Modal logics on rational Kripke structures

Wilmari Bekker

Student Number: 0616713N

Supervisor: Associate Professor Valentin Goranko

Masters Dissertation

School of Mathematics
Faculty of Science
University of the Witwatersrand
Johannesburg
South Africa

November 2007

Declaration

I declare that this dissertation is my own, unaided work. It is being submitted for the Degree of Master of Science in the University of the Witwatersrand, Johannesburg. It has not been submitted before for any degree or examination in any other University.

Wilmari Bekker

26^{th} day of November 2007, Johannesburg

Acknowledgments

I would to thank my supervisor, Valentin Goranko, for his instruction and guidance. He has made many opportunities available to me and has pushed me to make full use of them. Some of the work in this dissertation has been co-authored by Valentin Goranko and myself.

Furthermore, I would like to thank the National Research Foundation of South Africa for the funding provided during 2006 which made it possible to focus on my studies.

Many thanks to Arnaud Carayol, Christophe Morvan and Stephane Demri for some useful remarks and suggestions.

I would like to thank all my friends and family for their endless support and encouragement. In particular, I would like to thank my husband for his support and useful insight: "Diamonds belong on a ring in a Box." Thank you Grant.

Finally I give thanks to my Lord Jesus for giving me the talent and perseverance to see this study through.

Abstract

This dissertation is a contribution to the study of infinite graphs which can be presented in a finitary way. In particular, the class of rational graphs is studied. The vertices of a rational graph are labeled by a regular language in some finite alphabet and the set of edges of a rational graph is a rational relation on that language. While the first-order logics of these graphs are generally not decidable, the basic modal and tense logics are.

A survey on the class of rational graphs is done, whereafter rational Kripke models are studied. These models have rational graphs as underlying frames and are equipped with rational valuations. A rational valuation assigns a regular language to each propositional variable. I investigate modal languages with decidable model checking on rational Kripke models. This leads me to consider regularity preserving relations to see if the class can be generalised even further. Then the concept of a graph being rationally presentable is examined - this is analogous to a graph being automatically presentable. Furthermore, some model theoretic properties of rational Kripke models are examined. In particular, bisimulation equivalences between rational Kripke models are studied.

I study three subclasses of rational Kripke models. I give a summary of the results that have been obtained for these classes, look at examples (and non-examples in the case of automatic Kripke frames) and of particular interest is finding extensions of the basic tense logic with decidable model checking on these subclasses.

An extension of rational Kripke models is considered next: omega-rational Kripke models. Some of their properties are examined, and again I am particularly interested in finding modal languages with decidable model checking on these classes.

Finally I discuss some applications, for example bounded model checking on rational Kripke models, and mention possible directions for further research.

Contents

List of Figures

Chapter 1

Introduction

Verification of models with infinite state spaces using algorithmic symbolic model checking techniques has attracted considerable attention in recent years. Verifying safety, liveness, reachability, and other such properties, specified in some formal language, for computer programs is one of the main contributers that sparks interest in this field. There are many examples of infinite graphs stemming from computer science. At least five such 'sources of infinity' have been identified by Esparza in [38]:

1. Data manipulation: this includes unbounded counters, integer variables, lists, trees, more general pointer structures, etc.

2. Control structures: for example the dynamic creation of processes or unbounded call stacks.

3. Asynchronous communication: unbounded FIFO queues serves as an example.

4. Parameters: the parameterisation describing generic systems for example the number of processes, gates or buffers, or the duration of delays.

5. Real time: timing constraints in real-time systems with discrete or dense domains.

Verification is not possible on all infinite structures. It is not always possible to get a handle on an infinite structure which enables algorithmic verification and therefore verifying properties specified in some formal language is often undecidable. In [17] Blumensath and Grädel mention two fundamental conditions which must be satisfied by infinite structures on which verification is to be done :

1. The structure must be representable in a finite way.

2. The model checking problem of a relevant logic must be decidable on the domain of the structure.

One important development in infinite state verification has been *Caucal's hierarchy* [28], consisting of structures with decidable monadic second-order theories. In fact, the formal language in which the properties are specified here is very expressive. Reachability, liveness, safety, etc. properties can all be specified in second-order logic. Verifying these properties on graphs in Caucal's hierarchy is therefore decidable.

Another very successful approach to infinite state verification is based on representation of sets of states and transitions by means of various automata. The task is then to identify large and natural classes of (possibly) infinite structures which are representable by finite automata. Once such a class has been identified, formal languages that are sufficiently expressive and for which verification is decidable on the class of structures, need to be identified.

In [69] Morvan describes the use of various automata to represent infinite structures as an internal characterisation of the structure as opposed to an external characterisation where the structure (up to isomorphism) of the system is produced.

Various automata-based techniques for model checking are based on the use of finite machines to present an infinite structure. Some examples include model checking of linear and branching-time temporal logics on finite transition systems [90, 59], regular model checking [20], pushdown systems [19, 92, 40], and automatic structures [54, 16]. The temporal logics are sufficiently expressive for various reachability properties in most of the studied cases of infinite-state symbolic model checking. Unfortunately, each of these classes of models are relatively restricted.

In [70] Morvan studies a large and natural class of infinite graphs . The transition relations of rational graphs can be presented by finite asynchronous automata which are called transducers in the sequel. Automata-based techniques for model checking can therefore be applied to rational graphs. To this end, rational Kripke models are introduced in [46]. Rational Kripke models are based on rational graphs and every atomic proposition is evaluated is a regular set of states.

In this dissertation I study the class of rational Kripke models, on which global model checking of the basic tense[1] logic K_t (with forward and backward one step modalities) and some hybrid extensions thereof, is decidable. The language of K_t is sufficient for expressing local properties, in particular pre-conditions and post-conditions, but not reachability properties.

Kesten et al [53] have formulated the following minimal requirements for an assertional language $\mathcal{L}$ to be adequate for symbolic model checking:

1. The property to be verified and the initial conditions (i.e., the set of initial states) should be expressible in $\mathcal{L}$.

2. $\mathcal{L}$ should be effectively closed under the boolean operations, and should possess an algorithm for deciding equivalence of two assertions.

[1]The term 'tense' rather than 'temporal' is used, to emphasize that the accessibility relation is not assumed transitive, as in a usual flow of time.

3. There should exist an algorithm for constructing the predicate transformer *pred*, where $pred(\phi)$ is an assertion characterizing the set of states that have a successor state satisfying ϕ.

The first condition depends on the property to be verified while the set of initial states is usually assumed to be at least regular, if not a singleton, and can be represented by a special modal constant S. K is clearly closed under boolean operations if the equivalence is assumed to be with respect to the model on which the verification is being done. Finally, $pred(\phi) = \langle R \rangle \phi$ and an algorithm constructing it is presented in this dissertation. Thus, the basic modal logic K is the minimal natural logical language satisfying these requirements, and hence it suffices at least for specification of pre-conditions verified over regular sets of states. The tense extension K_t is the basic adequate logic for specifying local properties of transition systems since it enables specification of post-conditions as well.

Furthermore, the basic tense logic K_t can easily be extended to very expressive languages: reachability between two given states can be expressed in hybrid extensions of K_t, while reachability, liveness, safety, etc. properties can be expressed in linear- and branching-time temporal logics. The study of these logics has become an increasingly important area of research in recent years.

There are a number of ways to deal with the inability of K_t to express important properties like reachability. One way is to consider model constructions which preserve the truth of properties specified in one of the more expressive extensions of K_t. The other is to consider subclasses of the rational Kripke models.

Rational graphs have not been studied widely in the literature. The decidability of model checking of K_t (and extensions of it) in rational Kripke structures, however, warrants a closer look these structures. This is exactly the aim of this dissertation, while keeping model checking of modal languages in mind.

This dissertation is organised as follows: Chapter 2 contains all the necessary definitions and background theory, starting with the abstract structures monoids. Languages and relations which can be finitely presented as well as the machines used to present them are then studied. Basic modal logics is discussed briefly and the definition of the model construction bisimulation is revisited. Lastly Petri nets are presented. These are graphs originally introduced to solve concurrency problems in systems.

In Chapter 3 a summary of results on rational graphs is given. Rational Kripke models are studied in more detail for the remainder of that chapter. I consider a number of model checking tasks on these structures and show that local an global model checking K_t are decidable. The complexity of these results are then studied. The decidability of these model checking tasks then gives rise to a class of relations called regularity preserving relations. These relations are studied and as a result two types of two-sorted algebras find new importance. Some model theoretic results of rational Kripke models are presented and in particular the decidability of bisimulation equivalence between rational Kripke models and frames is examined.

Next, I study three subclasses of rational Kripke models in Chapter 4. Automatic Kripke models (these are Kripke models based on automatic graphs), rational Kripke trees (here the rational graph on which the model is based is acyclic and connected) and finally, monotonous rational Kripke models. Some model checking tasks are discussed for each of these subclasses.

In Chapter 5 I study the extension of rational Kripke models to ω-rational Kripke models. This extension forms an even more general class of structures.

In conclusion, some potential applications are mentioned and some open problems are discussed in Chapter 6.

Chapter 2

Preliminaries

2.1 Monoids

Recognisability and rationality of sets and relations are defined over the algebraic structures called monoids – a subclass of semigroups. In this section I present the relevant definitions.

Definition 2.1.1 ([7, 80]) *A **semigroup** is a structure $\mathcal{S} = (M, \bullet)$ where M is a set called the* **carrier set** *and $\bullet$ is an associative binary operation on M (in future referred to as the semigroup product or simply the product), i.e. $m_1 \bullet (m_2 \bullet m_3) = (m_1 \bullet m_2) \bullet m_3$ holds for all $m_1, m_2, m_3 \in M$.*

Definition 2.1.2 ([7, 36, 80]) *Let $\mathcal{S} = (M, \bullet, 1_M)$ where $(M, \bullet)$ is a semigroup and $1_M \in M$. If 1_M is such that for all $m \in M$ it holds that $1_M \bullet m = m \bullet 1_M = m$, then $\mathcal{S}$ is called a **monoid**, and the element 1_M is called a **unit**.*

The binary relation $\bullet$ can be extended to operate on subsets of the domain. If $\mathcal{S} = (M, \bullet, 1_M)$ is a monoid and $\Sigma, \Gamma \subseteq M$, then the product of Σ and Γ is the set:

$$\Sigma \bullet \Gamma = \{u \in M | \exists x \in \Sigma \; \exists y \in \Gamma : u = x \bullet y\}$$

The product of a subset Σ with itself will be denoted by $\Sigma^2 = \Sigma \bullet \Sigma$. Setting $\Sigma^0 = \{1_M\}$ it follows that $\Sigma^n = \Sigma \bullet \Sigma^{n-1} \; \forall n \geq 1$.

Definition 2.1.3 ([7, 36, 80]) *A **sub-semigroup (submonoid)** of a semigroup (respectively monoid) $\mathcal{S} = (M, \bullet)$ ($\mathcal{S} = (M, \bullet, 1_M)$) is a subset $\Sigma \subseteq M$ such that $\Sigma^2 \subseteq \Sigma$ (respectively $1_M \in \Sigma$ and $\Sigma^2 \subseteq \Sigma$).*

The least sub-semigroup and submonoid, with respect to set inclusion, that contains a subset $\Sigma \subseteq M$ are the sets Σ^+ and Σ^* respectively defined by

$$\Sigma^+ = \bigcup_{n \geq 1} \Sigma^n \qquad \Sigma^* = \bigcup_{n \geq 0} \Sigma^n$$

A monoid $\mathcal{S} = (M, \bullet, 1_M)$ is generated by a subset $\Sigma \subseteq M$, if $M = \Sigma^*$. If the subset Σ is finite then $\mathcal{S}$ is said to be **finitely generated**.

Definition 2.1.4 ([7, 36, 80]) *A monoid $\mathcal{S} = (M, \bullet, 1_M)$ is **free** if there is a non-empty subset $\Sigma \subseteq M$ such that $M = \Sigma^*$ and every element in M can either be uniquely expressed as the product of elements in Σ or is inexpressible as such a product.*[1] *If Σ is finite, $\mathcal{S}$ is said to be a **finitely generated free monoid**.*

Two or more monoids can now be used to define product monoids.

Definition 2.1.5 ([80]) *Let $\mathcal{S}_1 = (M_1, \bullet_1, 1_{M_1})$ and $\mathcal{S}_2 = (M_2, \bullet_2, 1_{M_2})$ be two monoids. The **cross-product monoid** of $\mathcal{S}_1$ and $\mathcal{S}_2$ is the monoid $(M_1 \times M_2, *, (1_{M_1}, 1_{M_2}))$ where $*$ is the component-wise application of the monoid products, i.e. if $(a, b), (c, d) \in M_1 \times M_2$ then $(a, b) * (c, d) = (a \bullet_1 c, b \bullet_2 d)$.*

In conclusion of this section, morphisms are defined.

Definition 2.1.6 ([7, 36, 80]) *Let $\mathcal{S}_1$ and $\mathcal{S}_2$ be monoids. A **monoid morphism** (just morphism for short) is a function $\alpha : M_1 \to M_2$, where M_1 and M_2 are the respective domains and $\bullet_1$ and $\bullet_2$ the product operations of $\mathcal{S}_1$ and $\mathcal{S}_2$ respectively, which satisfies:*

1. $\alpha(u \bullet_1 v) = \alpha(u) \bullet_2 \alpha(v)$
2. $\alpha(1_M) = 1_{M'}$

2.2 Languages

For any finite set Σ, there is a finitely generated free monoid where the product operation of the monoid is the concatenation of elements, i.e. for $(\Sigma^*, \bullet, 1_{\Sigma^*})$ the elements of Σ^* are tuples of elements from Σ:

$$u \in \Sigma^* \longrightarrow \text{there exists } n \in \mathbb{N} \text{ such that } u = x_1 x_2 \cdots x_n \text{ where } x_1, x_2, \ldots, x_n \in \Sigma$$

Furthermore, if $u, v \in \Sigma^*$ where $u = x_1 x_2 \cdots x_n$ and $v = y_1 y_2 \cdots y_m$ and $n, m \in \mathbb{N}$ then $u \bullet v = x_1 x_2 \cdots x_n y_1 y_2 \cdots y_m = uv$. (The $\bullet$ sign is omitted since it is known that the product operation is the concatenation of the elements.) The unit 1_{Σ^*} of this monoid will then be the 0-tuple denoted by ϵ.

Under these circumstances the set Σ can be viewed as an alphabet. Words in an alphabet Σ are finite sequences of elements from the alphabet which are exactly the

[1] Clearly the unit cannot be an element of Σ, for suppose the element 1_M belonged to Σ and let $x \in \Sigma$ be an arbitrary element of Σ, then $x \in M$ can be expressed as $1_M \bullet x = x = x \bullet 1_M$, which is not unique. Therefore $1_M \notin \Sigma$.

elements of the monoid Σ^* as defined above and the unit is the empty word. A language over an alphabet consists of words over an alphabet on which a number of restrictions are placed. Therefore, a language over Σ is a subset of Σ^*.

I will use Σ and Γ to denote alphabets, words will be denoted by u, v, w and x, y, z will be used to denote letters in the alphabet.

A hierarchy of four families of languages were introduced by Chomsky [66, 84] as syntactic models of natural language. These four families are the **regular languages**, **context-free languages**, **context-sensitive languages** and **recursively enumerable languages** - each family strictly including the previous. I will discuss two of these families in this dissertation, namely the context-free and regular languages. Special notice will be taken of the latter since it will be particularly useful when doing model checking. The definitions of the other two families of languages can be found in [84].

The strings of a language are formally generated by a set of rules, called a **grammar**. By placing different restrictions on the rules of grammars, different classes of languages can be generated.

Definition 2.2.1 ([7, 66, 84]) *A **context-free grammar** is a quadruple $\mathcal{G} = \langle V, \Sigma, P, S\rangle$ where V is a finite set of variables, the set Σ is a finite alphabet, $V \cap \Sigma = \emptyset$, the symbol S belongs to V and P is a finite set of rules belonging to $\Gamma \times (\Gamma \cup \Sigma)^*$.*

The elements of Σ are called **terminal symbols** *since they form the indecomposable elements used to construct strings in the languages and S is called the* **start symbol**.

Example 2.2.2 Let $\mathcal{G} = \langle V, \Sigma, P, S\rangle$ be defined by

1. $V = \{A, B\}$
2. $\Sigma = \{x, y\}$
3. $P = \{A \to xB|y;\ \ B \to Ay\}$
4. $S = A$

The '|' sign in the rule referrers to a choice, i.e. the variable A can be replaced by either the symbols xB or by the terminal symbol y. ◁

A string (word) is generated by a grammar if it can be produced by a finite succession of rules in the grammar starting with the designated starting symbol in the grammar. The set of all such words generated by a grammar is the language generated by the grammar.

Definition 2.2.3 ([7, 66, 84]) *A language $\mathcal{L}$ over some alphabet Σ is a **context-free language** if it is generated by some context-free grammar $\mathcal{G}$.*

Example 2.2.4 Consider the language $\mathcal{L} = \{x^n y^{n+1} | n \geq 0\}$. This is the language generated by the context-free grammar in Example 2.2.2. ◁

Definition 2.2.5 ([66]) *Let $\mathcal{G} = \langle V, \Sigma, P, S\rangle$ be a grammar where Γ, Σ and S are defined as before, and where the rules in P are all in one of the following forms. For A,$B \in \Gamma$ and $x \in \Sigma$*

1. $A \rightarrow x$
2. $A \rightarrow xB$
3. $A \rightarrow \epsilon$

*where $B \neq A$. Under these conditions the grammar $\mathcal{G}$ is a **right regular grammar**.*

The grammar defined above is also called a right-linear grammar. **Left regular grammars** can be defined in a similar way.

Observe that the rule $B \rightarrow Ay$ in Example 2.2.2 is not in the right form for the grammar to be right-regular. In fact, the language generated by this grammar, as given in Example 2.2.4 cannot be generated by either a left- or right-regular grammar.

Example 2.2.6 Let $\mathcal{G} = \langle V, \Sigma, P, S\rangle$ be a grammar with V, Σ and S defined in the same way as in Example 2.2.2. Furthermore, let $P = \{A \rightarrow x|xB \quad B \rightarrow yA\}$, then $\mathcal{G}$ is a regular grammar. ◁

Definition 2.2.7 ([66]) *A language $\mathcal{L}$ over some alphabet Σ is a **regular language** if it can be generated by some left or right regular grammar $\mathcal{G}$.*

Example 2.2.8 $\{(xy)^n x|n \geq 0\}$ is the regular language generated by the regular grammar shown in Example 2.2.6. ◁

2.3 Automata and Transducers

In this section classes of machines are defined. These machines will be used as language recognisers.

Definition 2.3.1 ([80]) *An* **abstract finite automaton** *is a quadruple $\mathcal{A} = \langle Q, M, q^0, F, \delta\rangle$ where Q is a finite set of states, $F \subseteq Q$ is a set of accepting states, $q^0 \in Q$ is a unique initial state, M is the domain of some monoid $\mathcal{S} = (M, \bullet, 1_M)$ and $\delta : Q \times M \rightarrow Q$ is the next state function satisfying:*

1. $\delta(q, 1_M) = q$ *holds for all* $q \in Q$
2. $\delta(q, m_1 \bullet m_2) = \delta(\delta(q, m_1), m_2)$ *for all* $q \in Q$ *and for all* $m_1, m_2 \in M$.

When a finitely generated free monoid is considered, the definition reads as given below.

Definition 2.3.2 ([7, 36, 80, 84]) *Given a finite set of states Q, a finite alphabet Σ, a state $q^0 \in Q$, a subset $F \subseteq Q$ and a relation $\delta : Q \times \Sigma \to Q$, then the quadruple $\mathcal{A} = \langle Q, \Sigma, q^0, F, \delta \rangle$ is a* ***finite deterministic automaton***

Given an automaton $\mathcal{A} = \langle Q, \Sigma, q^0, F, \delta \rangle$, then a sequence $q_1 \overset{x_1}{\to} q_2 \overset{x_2}{\to} \cdots \overset{x_n}{\to} q_{n+1}$ is called a **path** if $q_j \in Q$ for $j = 1, 2, \ldots, n+1$ and $x_1, x_2, \ldots, x_n \in \Sigma$ such that $q_{i+1} \in \delta(q_i, x_i)$ for $i = 1, 2, \ldots, n$. If $q_1 = q^0$ then the path is called a **run** of the automaton.

Definition 2.3.3 ([7, 36, 80, 84]) *A word $u \in \Sigma^*$ is* **accepted** *by $\mathcal{A} = \langle Q, \Sigma, q^0, F, \delta \rangle$ if and only if there exists a run $q_1 \overset{x_1}{\to} q_2 \overset{x_2}{\to} \cdots \overset{x_n}{\to} q_{n+1}$ such that $q_{n+1} \in F$, $n \in \mathbb{N}$ and $u = x_1 x_2 \cdots x_n$ (Thus the run is labeled by the word u). Such a run is called an* **accepting run**. *The set of all words accepted by $\mathcal{A}$, denoted by $\mathcal{L}(\mathcal{A})$, is said to be the language* **recognised** *by $\mathcal{A}$.*

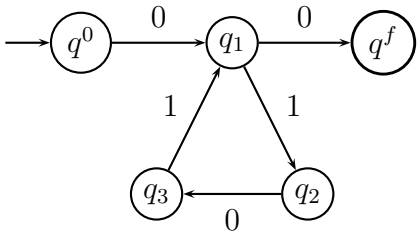
Instead, the next state relation can also be defined such that $\delta : Q \times \Sigma \to \mathcal{P}(Q)$. An automaton with a next state relation is a **nondeterministic finite automaton**. Another variation of an automaton would be to define the the next state function on words instead of letters, i.e. $\delta : Q \times \Sigma^* \to Q$.

Example 2.3.4 Let $\mathcal{A} = \langle Q, \Sigma, q^0, F, \delta \rangle$ be a nondeterministic finite automaton with

1. $Q = \left(q^0, q_1, q_2, q_3, q^f\right)$,
2. $\Sigma = \{0, 1\}$,
3. $F = \left\{q^f\right\}$, and
4. $\delta = \left\{\left(q^0, 0, q_1\right), \left(q_1, 1, q_2\right), \left(q_2, 0, q_3\right), \left(q_3, 1, q_1\right), \left(q_1, 0, q^f\right)\right\}$.

$\mathcal{A}$ can now be represented by a graph in the following way:

Figure 2.1: The automaton $\mathcal{A}$ recognising the regular set $0(101)^* 0$.



The word $u = 01011010$ is recognised by $\mathcal{A}$ since there is an accepting run in $\mathcal{A}$ labeled by u. ◁

Finite automata can now be extended in such a way that the resulting class of machines are able to recognise a more general class of languages.

Definition 2.3.5 ([84]) *The tuple* $\mathcal{M} = \langle Q, \Sigma, \Gamma, q^0, F, \delta \rangle$ *is a* ***pushdown automaton*** *when* Σ *and* Γ *are finite disjoint alphabets, called the input and stack alphabets respectively,* Q *is a set of states,* $q^0 \in Q$ *a unique starting state,* $F \subseteq Q$ *a set of final states, and* $\delta \subseteq Q \times (\Sigma \cup \{\epsilon\}) \times (\Gamma \cup \{\lambda\}) \times Q \times (\Gamma \cup \{\lambda\})$ *is a finite transition relation. (*ϵ *is the empty word in* Σ^* *and* λ *the empty word in* Γ^**.)*

Let $u \in \Sigma^*$, then u is accepted by the pushdown automaton $\mathcal{M} = \langle Q, \Sigma, \Gamma, q^0, F, \delta \rangle$ if, starting with an empty stack at the initial state q^0 and then reading u letter by letter, following the transition relation δ step by step, the process ends with an empty stack at a final state.

Example 2.3.6 $\mathcal{M} = \langle Q, \Sigma, \Gamma, q^0, F, \delta \rangle$ is a pushdown automaton when

1. $Q = \left\{q^0, q^f\right\}$
2. $\Sigma = \{0, 1\}$
3. $\Gamma = \{A\}$
4. $F = \left\{q^f\right\}$; and
5. $\delta = \left\{\left(q^0, 0, \lambda, q^0, A\right), \left(q^0, 1, A, q^0, A\right), \left(q^0, 1, A, q^f, \lambda\right)\right\}$

For example, the word 0111 is recognised by $\mathcal{M}$. ◁

The transition relation of a pushdown automaton can also be denoted as a function: $\delta : Q \times (\Sigma \cup \{\epsilon\}) \times (\Gamma \cup \{\lambda\}) \to \mathcal{P}(Q \times (\Gamma \cup \{\lambda\}))$. The relation δ in example 2.3.6 would then be denoted as follows:

$$\delta\left(q^0, 0, A\right) = \{[q_1, A]\}$$

$$\delta\left(q^0, 1, A\right) = \left\{\left[q^0, A\right], [q_f, \lambda]\right\}$$

A machine recognising pairs of words is required when cross-product monoids are considered.

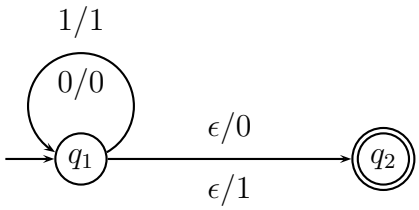
Definition 2.3.7 ([7, 36, 80]) *A tuple* $\mathcal{T} = \langle Q, \Sigma, \Gamma, q^0, F, \rho \rangle$ *is called a* ***transducer*** *where* Σ *and* Γ *are the input and output alphabets respectively,* Q *a set of states,* $q^0 \in Q$ *a unique starting state,* $F \subseteq Q$ *a set of accepting states and* ρ *the finite next-state relation satisfying:* $\rho \subseteq Q \times \Sigma^* \times \Gamma^* \times Q$*.*

A run and an accepting run of a finite transducer are defined similar to a run and an accepting run for an automaton.

Example 2.3.8 For $\mathcal{T} = \langle Q, \Sigma, \Gamma, q^0, F, \rho \rangle$ let:

1. $Q = \{q_1, q_2\}$;
2. $\Sigma = \{0, 1\} = \Gamma$;
3. $q^0 = q_1$;
4. $F = \{q_2\}$; and
5. $\rho = \{(q_1, 0, 0, q_1), (q_1, 1, 1, q_1), (q_1, \epsilon, 0, q_2), (q_1, \epsilon, 1, q_2)\}$

Figure 2.2: The transducer $\mathcal{T}$ which recognises pairs of words of either the form $(u, u0)$ or $(u, u1)$ where $u \in \Sigma^*$ and ux denotes the concatenation of the word u and the letter x.



In the representation of $\mathcal{T}$ there is only one edge between two states but that an edge may have more that one label, i.e. the edge represents multiple transitions.

$(0100, 01000)$ and $(0100, 01001)$ are examples of pairs of words recognised by $\mathcal{T}$. $\lhd$

2.4 Recognisable, rational, and regular sets

In this section I discuss two classes of subsets of arbitrary monoids. When the monoids are free and finitely generated by some alphabet, these are two classes of languages.

Definition 2.4.1 ([80]) *Let $\mathcal{M}_1 = (M_1, \bullet_1, 1_{M_1})$ be an arbitrary monoid and let $A_1 \subseteq M_1$. The set A_1 is called a **recognisable subset** of M_1 if there exists a finite monoid, say $\mathcal{M}_2 = (M_2, \bullet_2, 1_{M_2})$, a morphism $\alpha : M_1 \to M_2$ and a subset $A_2 \subseteq M_2$ such that $A_1 = \alpha^{-1}(A_2)$.*

Let $\mathsf{REC}(M)$ denote the set of all recognizable subsets of M. Another, perhaps more natural characterisation of recognisable sets is in terms of the machines that recognise them.

Theorem 2.4.2 ([7, 80]) *A subset $A \subseteq M$ of a monoid is recognisable if and only if it is recognised by some finite abstract automaton over M.*

Theorem 2.4.3 ([7, 80]) *The class $\mathsf{REC}(M)$ of recognisable subsets of some monoid $(M, \bullet, 1_M)$ is closed under the following operations:* ***union, intersection, complementation*** *and* ***set difference****. It is also closed under* ***inverse morphisms*** *and* ***isomorphisms****.*

The next family of subsets under consideration can be defined by induction.

Definition 2.4.4 ([7, 80]) *Again let $\mathcal{M} = (M, \bullet, 1_M)$ be an arbitrary monoid. The family of* ***rational subsets*** *of M, denoted by $\mathsf{RAT}(M)$, is the least family $\mathcal{R}$ of subsets of M such that:*

1. $\emptyset \in \mathcal{R}$;
   For all $x \in M$ $\{x\} \in \mathcal{R}$;
2. *If* $X, Y \in \mathcal{R}$ *then* $X \cup Y \in \mathcal{R}$ *and* $XY \in \mathcal{R}$;
3. *If* $X \in \mathcal{R}$ *then* $X^+ = \bigcup_{n \geq 1} X^n \in \mathcal{R}$.

The next theorem now sums up the closure properties of this class of subsets:

Theorem 2.4.5 ([7, 80]) *The class $\mathsf{RAT}(M)$ of rational subsets of some monoid $(M, \bullet, 1_M)$ is closed under the following operations:* ***union, the monoid product operation, Kleene's plus and star operations*** *and* ***morphisms****.*

Proposition 2.4.6 ([7]) *Any subset of a finite monoid is rational.*

Proposition 2.4.7 ([7]) *If $(M, \bullet, 1_M)$ is an arbitrary monoid, $X \in \mathsf{RAT}(M)$ and $Y \in \mathsf{REC}(M)$ then $X \cap Y \in \mathsf{RAT}(M)$.*

The relationship between the recognisable and rational sets of a finitely generated monoid were given by McKnight in 1964.

Theorem 2.4.8 (McKnight,[7]) *If $(M, \bullet, 1_M)$ is a finitely generated monoid, then $\mathsf{REC}(M) \subseteq \mathsf{RAT}(M)$.*

By imposing a stronger condition on the monoids considered, it can be proved that these two classes of subsets coincide.

Theorem 2.4.9 (Kleene,[7, 36, 80]) *If $(M, \bullet, 1_M)$ is a finitely generated free monoid, i.e. there exists a finite alphabet Σ such that $M = \Sigma^*$, then a subset $X \subseteq M$ is recognisable if and only if it is rational.*

Theorem 2.4.2 can now be rephrased since the automata now operate on words and therefore need not be in terms of abstract automata.

Theorem 2.4.10 ([7, 80]) *A language $X \subseteq \Sigma^*$ is rational/recognisable if and only it is accepted by some finite deterministic automaton.*

In fact, a language X is rational/recognisable if and only if it is accepted by some finite nondeterministic automaton [7]. The theorem above ensures that every rational subset is recognised by a finite deterministic automaton.

This class of subsets is a class of languages over the alphabet known as the regular languages over the alphabet. When working on finitely generated free monoids the terms 'rational', 'recognisable' and 'regular' can be used indistinctly. In the sequel I will use the term **regular** to emphasize that the monoid is finitely generated and free. A direct consequence of Kleene's theorem is that all the closure properties of recognisable and rational subsets of monoids carries over to the class of regular subsets.

Definition 2.4.11 ([84]) *Let $u \in \Sigma^*$ for some finite Σ, then the **reversal** u^R of u is the word obtained recursively by:*

1. *If $u = \epsilon$ then $u^R = \epsilon$;*
2. *If $u = vx$ for $v \in \Sigma^*$ and $x \in \Sigma$, then $u^R = xv^R$.*

Definition 2.4.12 ([7, 36]) *Let $X, Y \subseteq \Sigma^*$ for Σ finite. The **internal shuffle product** of X and Y is the set:*

$$X \sqcup Y \{u \in \Sigma^* | u = a_1 b_1 a_2 b_2 \cdots a_n b_n,\ a_i \in X,\ b_i \in Y\ \ (i = 1, 2, \ldots, n)\}$$

Theorem 2.4.13 *The class $\mathsf{REG}(\Sigma^*)$ of regular sets of Σ^* is closed under the following operations: **union, product, the Kleene star and plus operations, intersection, set difference, the reversal operation, regular substitution, morphism, inverse morphism** [7] and **the internal shuffle product** [36].*

Denote the **length** of a word (string) $u \in X$ by $|u|$. Regular sets also have the properties given below.

Theorem 2.4.14 (Pumping lemma for regular sets, [7, 84]) *Let X be a regular set recognised by the finite deterministic automaton $\mathcal{A}$ which has k states. If $u \in X$ is any word such that $|u| \geq k$, then u can be written in terms of some factors $w_1, w_2, w_3 \in \Sigma^*$, i.e. $u = w_1 w_2 w_3$ where $|w_1 w_2| \leq k$, $|w_2| > 0$ and $w_1 w_2^i w_3 \in X$ for all $i \geq 0$.*

The pumping lemma serves as a powerful tool to prove that languages are not regular. To this end it suffices to give one word in the languages which does not satisfy the pumping lemma.

Theorem 2.4.15 ([80]) *Suppose $X, Y \in \mathsf{REG}(\Sigma^*)$ is recognised by the finite deterministic automaton $\mathcal{A}$ where Σ is finite. Then it is decidable whether X is **empty**, **finite** or **infinite**. It is also decidable whether $X \subseteq Y$.*

2.5 Recognisable, and rational relations

The recognisable and rational subsets of cross-product monoids form useful classes of relations.

For monoids $(M, \bullet_1, 1_M)$ and $(M', \bullet_2, 1_{M'})$, the class $\mathsf{REC}(M \times M')$ is called the class of recognisable binary relations on $M \times M'$. Likewise a binary relation on $M \times M'$ is said to be rational if and only if it belongs to the class $\mathsf{RAT}(M \times M')$. Unless otherwise specified, all the relations in the sequel are assumed to be binary.

An alternative characterisation in terms of regular languages exists for recognisable relations.

Theorem 2.5.1 (Mezei, [7]) *Let $(M_1, \bullet_1, 1_{M_1})$ and $(M_2, \bullet_2, 1_{M_2})$ be two arbitrary monoids. A subset of the cross-product monoid $R \subseteq M_1 \times M_2$ is recognisable if and only if it is the union of finitely many sets of the form $X \times Y$ where $X \in \mathsf{REC}(M_1)$ and $Y \in \mathsf{REC}(M_2)$.*

Only the recognisable and rational subsets of the cross-product of two finitely generated free monoids will be considered in the sequel. Then rational relations (subsets) can be characterised in terms of the machines recognising them.

Theorem 2.5.2 ([7]) *Let Σ and Γ be finite alphabets. A relation $R \subseteq \Sigma^* \times \Gamma^*$ is rational if and only if it is recognised by a finite transducer.*

Moreover, the existence of a specific type of transducer recognising each rational relation is guaranteed.

Corollary 2.5.3 ([7]) *Any rational relation $R \subseteq \Sigma^* \times \Gamma^*$ can be recognised by a transducer $\mathcal{T} = \langle Q, \Sigma, \Gamma, q_i, F, \rho \rangle$ such that*

$$\rho \subseteq Q \times (\Sigma \cup \{\epsilon\}) \times (\Gamma \cup \{\epsilon\}) \times Q$$

and furthermore, F consists of a single state $q_f \neq q_i$, and $(p_1, u, v, p_2) \in \rho$ implies $p_1 \neq q_f$ and $p_2 \neq q_i$.

In 1968 Nivat produced a theorem giving four conditions equivalent to a relation being a rational relation.

Theorem 2.5.4 (Nivat, [74]) *Let Σ and Γ be finite alphabets. The following are equivalent:*

1. *$R \in \mathsf{RAT}(\Sigma^* \times \Gamma^*)$;*

2. *There exists an alphabet* Υ*, two morphisms* $\alpha_1 : \Upsilon^* \to \Sigma^*$*,* $\alpha_2 : \Upsilon^* \to \Gamma^*$ *and a regular language* $X \subseteq \Upsilon^*$ *such that*

$$R = \{(\alpha_1(u), \alpha_2(u)) \,|u \in X\}$$

3. *There exists an alphabet* Υ*, two alphabetic morphisms* $\beta_1 : \Upsilon^* \to \Sigma^*$*,* $\beta_2 : \Upsilon^* \to \Gamma^*$ *and a regular language* $X \subseteq \Upsilon^*$ *such that*

$$R = \{(\beta_1(u), \beta_2(u)) \,|u \in X\}$$

4. *There exists an alphabet* Υ*, two alphabetic morphisms* $\beta_1 : \Upsilon^* \to \Sigma^*$*,* $\beta_2 : \Upsilon^* \to \Gamma^*$ *and a local regular language* $Y \subseteq \Upsilon^*$ *such that*

$$R = \{(\beta_1(u), \beta_2(u)) \,|u \in Y\}$$

If $\Sigma \cap \Gamma = \emptyset$*, then (1) is equivalent to:*

5. *There exists a regular language* $X \subseteq (\Sigma \cup \Gamma)^*$ *such that*

$$R = \{(\pi_\Sigma(u), \pi_\Gamma(u)) \,|u \in S\}$$

where π_Σ *and* π_Γ *are the projections of* $(\Sigma \cup \Gamma)^*$ *onto* Σ^* *and* Γ^* *respectively.*

The last three conditions make use of alphabetic morphisms, local regular languages and projections. These concepts are not used in this study and therefore not defined here. Definition can however be found in [7, 80]. Nivat's theorem can now be used to produce an iteration lemma for rational relations, which is similar to the pumping lemma for regular sets.

Lemma 2.5.5 (Iteration lemma for rational relations, [7]) *If* $R \in \mathsf{RAT}(\Sigma^* \times \Gamma^*)$ *where* Σ *and* Γ *are finite, then there exists an integer* $k \geq 1$ *such that if* $(u, v) \in R$ *with* $|u| + |v| \geq k$*, then* (u, v) *has a factorisation:*

$$(u, v) = (a_1, a_2)(b_1, b_2)(c_1, c_2)$$

for $a_1, b_1, c_1 \in \Sigma^*$ *and* $a_2, b_2, c_2 \in \Gamma^*$ *such that*

1. $0 \leq |b_1| + |b_2| \leq k$*;*
2. $(a_1, a_2)(b_1, b_2)^*(c_1, c_2) \subseteq R$*.*

The cross-product monoid of two finitely generated monoids is also finitely generated and therefore it follows from Theorem 2.4.8 that every recognisable relation is also rational. The closure properties described in theorem 2.4.3 and theorem 2.4.5 naturally also apply to the binary relations $\mathsf{REC}(\Sigma^* \times \Gamma^*)$ and $\mathsf{RAT}(\Sigma^* \times \Gamma^*)$ respectively.

Lemma 2.5.6 ([51]) *If* $X \in \mathsf{RAT}(\Sigma^*)$ *then* $I_X = \{(u,u) \,|u \in X\} \in \mathsf{RAT}(\Sigma^* \times \Sigma^*)$.

A rational relation $R \subseteq \Sigma^* \times \Sigma^*$ can also be seen as a function from Σ^* into the power set $\mathcal{P}(\Sigma^*)$, and is then called a rational transduction. The domain and the range of R are defined by:

$$dom(R) = \{u \in \Sigma^* | R(u) \neq \emptyset\}$$

$$ran(R) = \{v \in \Sigma^* | \exists u \in \Sigma^* : v \in R(u)\}$$

Nivat's theorem can be reformulated for rational transductions. The following useful results are direct consequences of Nivat's theorem

Corollary 2.5.7 ([7]) *If* $R \in \mathsf{RAT}(\Sigma^* \times \Gamma^*)$ *then* $dom(R) \in \mathsf{RAT}(\Sigma^*)$ *and* $ran(R) \in \mathsf{RAT}(\Gamma^*)$.

Corollary 2.5.8 ([7]) *If* $R \in \mathsf{RAT}(\Sigma^* \times \Gamma^*)$ *and* $X \in \mathsf{RAT}(\Sigma^*)$ *then* $R(X) \in \mathsf{RAT}(\Gamma^*)$.

In fact, in the theorem above the first monoid need not be free - only the second monoid needs to be free for the result to hold ([36]).

Theorem 2.5.9 (Elgot and Mezei, [37]) *Let* Σ, Γ *and* Υ *be finite alphabets,* $R_1 \in \mathsf{RAT}(\Sigma^* \times \Gamma^*)$ *and* $R_2 \in \mathsf{RAT}(\Gamma^* \times \Upsilon^*)$. *Then the relations* $R_1 \circ R_2 \subseteq \Sigma^* \times \Upsilon^*$ *is rational.*

In general the composition of rational relations does not preserve rationality, but the following result holds.

Proposition 2.5.10 (Composition theorem, [7]) *Let* M_1 *and* M_2 *be the carrier sets of arbitrary monoids and let* Σ *be a finite alphabet. If* $R_1 \in \mathsf{RAT}(M_1 \times \Sigma^*)$ *and* $R_2 \in \mathsf{RAT}(\Sigma^* \times M_2)$, *then* $R_1 \circ R_2 \in \mathsf{RAT}(M_1 \times M_2)$ - *where* $\circ$ *denotes the compositions of the relations which can be seen as functions into the power set of the second monoid.*

Since Elgot and Mezei's theorem is used to prove this proposition, the condition that the two monoids are finitely generated and free is necessary.

In conclusion I present a number of decidability results for recognisable, and rational relations.

Definition 2.5.11 ([51]) *A rational relation is* **length-bounded** *if there is some* $k \in \mathbb{N}$ *such that* $|v| - k \leq |u| \leq |v| + k$ *for every* $(u,v) \in R$

Theorem 2.5.12 ([80]) *Let* $R_1, R_2 \in \mathsf{RAT}(\Sigma^* \times \Gamma^*)$ *for finite alphabets* Σ *and* Γ. *Then it is decidable whether or not*

1. R_1 *is finite;*

2. R_1 *is empty;*

Theorem 2.5.13 ([80]) *Let* Σ *and* Γ *be finite alphabets containing at least two letters and* $R_1, R_2 \in \mathsf{RAT}(\Sigma^* \times \Gamma^*)$ *then it is undecidable whether or not*

1. $R_1 \cap R_2 = \emptyset$*;*
2. $R_1 \subset R_2$*;*
3. $R_1 = R_2$*;*
4. $R_1 = \Sigma^* \times \Gamma^*$*;*
5. R_1 *is co-finite;*
6. R_1 *recognisable.*

On the other hand, if $R_1, R_2 \in \mathsf{REC}(\Sigma^* \times \Gamma^*)$ *the decision problems listed above become decidable.*

Finally:

Theorem 2.5.14 ([51]) *Given a finite transducer for a rational relation* R*, it is recursively undecidable whether* R *is transitive, symmetric, reflexive or an equivalence relation. On the other hand it is decidable whether or not* R *is a length-bounded rational equivalence relation.*

2.6 Modal logics

In order to be able to adequately describe and reason about relational structures, something needs to be added to propositional languages which will be able to access the necessary information of the relational structure. By adding modalities (modal operators) to propositional languages, local properties of the structures can be described.

2.6.1 Syntax

Definition 2.6.1 ([12]) *Let* $\Phi = \{p_1, p_2, \ldots, p_n\}$ *be a set of propositions. A formula* φ *is in the* ***basic modal language*** **K** *if it satisfies the rule:*

$$\varphi := p | \bot | \neg\varphi | \phi \vee \varphi | \Diamond\varphi,$$

where $\Diamond$ *is a unary operator called the* **diamond operator***, and* p *is an arbitrary element of* Φ*.*

A dual operator, called the box operator denoted by $\Box$, can be defined for the diamond operator by setting $\Box\varphi := \neg\Diamond\neg\varphi$. Furthermore the connectives conjunction, implication and bi-implication are abbreviations for the following formulas respectively: $\varphi \wedge \phi := \neg(\neg\varphi \vee \neg\phi)$, $\varphi \rightarrow \phi := \neg\varphi \vee \phi$ and $\varphi \leftrightarrow \phi := (\varphi \rightarrow \phi) \wedge (\phi \rightarrow \varphi)$. The constant true, $\top$ will abbreviate $\neg\bot$. In the sequel a priority order will be imposed on the connectives, as for propositional logic, and unnecessary parentheses will be omitted. The priority order will be:

1. $\neg$, $\Box$ and $\Diamond$
2. $\vee$ and $\wedge$
3. $\rightarrow$
4. $\leftrightarrow$

Example 2.6.2 The following are examples of modal formulas:

1. $\Box(\Box p \rightarrow p) \rightarrow \Box p$
2. $\neg\Diamond(p \vee q) \vee (\Diamond p \vee \Diamond q)$
3. $\Diamond\Box p \rightarrow \Box\Diamond p$

$\lhd$

Consider the formula $\underbrace{\Diamond\Diamond\cdots\Diamond}_{n}\varphi$ where the diamond operator is applied $n \geq 0$ times. It can be abbreviated by defining the operator $\Diamond^n = \underbrace{\Diamond\Diamond\cdots\Diamond}_{n}$. Furthermore, let $\Diamond^*$ denote the reachability operator.

K can be extended by adding various modal operators.

Definition 2.6.3 *[10, 91] A formula φ is in the* **basic tense language $\mathbf{K_t}$** *if it satisfies the rule:*

$$\varphi := p|\bot|\neg\varphi|\phi \vee \varphi|\Diamond\varphi|\Diamond^{-1}\varphi,$$

where $p \in \Phi$ and $\Diamond^{-1}$ is a unary modal operator called the **backwards diamond** *operator.*

The symbols $\langle R\rangle$ and $\langle R^{-1}\rangle$ may also be used instead of $\Diamond$ and $\Diamond^{-1}$. The generalization of what follows to the case of languages and models with many relations is straightforward.

Temporal logics (see [63, 91]) are also very important extensions of the basic modal logic, used to express successive events along time. The two main frameworks within

which temporal logics have been studied are **linear-time** and **branching-time**. Linear-time temporal logic expresses properties over a single path in a model. The path is often referred to as a time line while branching-time temporal logic is the extension of linear-time temporal logics with path quantifiers A and E expressing 'along all paths' and 'along some path'. These extensions of the basic modal logic are very expressive and important properties such as liveness and safety properties can be expressed in them. Bounded model checking (see chapter 6) of rational Kripke models will provide a method of partial model checking of temporal logic formulas.

2.6.2 Semantics

The languages defined above still have no context or interpretation. Modal languages must be interpreted in **relational structures** (relational structures are tuples containing a nonempty set called the **universe** and a number of **relations** on that universe). This is called relational semantics. There are two levels at which modal languages can be interpreted. The first is the level of **frames** and the second the level of **models**.

Definition 2.6.4 *A **Kripke frame** is a pair $\mathcal{F} = (S, R)$ such that S is non-empty, called the universe, and R is a binary relation on S, i.e. $R \subseteq S \times S$.*

Definition 2.6.5 *$\mathcal{M} = (S, R, V)$ is a **Kripke model** if (S, R) is a Kripke frame and $V : \Phi \to \mathcal{P}(S)$ is a function that assigns a subset of the universe to each propositional variable.*

A model $\mathcal{M}$ can also be written as a pair $(\mathcal{F}, V)$ where $\mathcal{F} = (S, R)$ is a Kripke frame. Then $\mathcal{M}$ is said to be based on $\mathcal{F}$. Furthermore, let $L : S \to \mathcal{P}(\Phi)$ be the function defined by: $L(x) = \{p \in \Phi | x \in V(p)\}$ where $x \in S$.

Example 2.6.6 Let $S = \mathbb{N}$ and if $x, y \in \mathbb{N}$ then xRy if and only if $y = x + 2$. Then the relational structure $\mathcal{F} = (S, R)$ is a Kripke frame.

Suppose $\Phi = \{p, q\}$ and define $V : \Phi \to \mathcal{P}(S)$ to be the function:

$$\begin{aligned} V(p) &:= \{x \in \mathbb{N} | x = 2k,\ k \in \mathbb{N}\} \\ V(q) &:= \{y \in \mathbb{N} | y = 2k + 1,\ k \in \mathbb{N}\} \end{aligned}$$

Then $\mathcal{M} = (\mathcal{F}, V)$ is based on $\mathcal{F}$. ◁

It is not enough to just define relational structures. How to interpret these modal languages on Kripke structures must also be defined.

Definition 2.6.7 ([12]) *Let $\mathcal{M} = (S, R, V)$ be a Kripke model and let $s \in S$. Then **satisfaction** of a formula φ in K_t in $\mathcal{M}$ at s is inductively defined by:*

$$\begin{aligned}
\mathcal{M}, s \models p &\iff s \in V(p), \text{ for } p \in \Phi \\
\mathcal{M}, s \models \bot &\quad \text{never} \\
\mathcal{M}, s \models \neg\varphi &\iff \text{it is not the case that } \mathcal{M}, s \models \varphi \\
\mathcal{M}, s \models \varphi \vee \phi &\iff \mathcal{M}, s \models \varphi \text{ or } \mathcal{M}, s \models \phi \\
\mathcal{M}, s \models \Diamond\varphi &\iff \text{there is some } t \in S \text{ such that } sRt \text{ and } \mathcal{M}, t \models \varphi \\
\mathcal{M}, s \models \Diamond^{-1}\varphi &\iff \text{there is some } t \in S \text{ such that } tRs \text{ and } \mathcal{M}, t \models \varphi
\end{aligned}$$

Note that the relation R is not assumed transitive, and therefore the language of K_t cannot express R-reachability properties. On the otherhand, K_t can express local properties.

Example 2.6.8 Let $\mathcal{M}$ be the model defined in Example 2.6.6 and let $\varphi = \Diamond p$. Then $\mathcal{M}, 2 \models \varphi$ and $\mathcal{M}, 3 \not\models \varphi$. ◁

A single R (respectively, R^{-1}) step is made by $\Diamond$ ($\Diamond^{-1}$), while $\Diamond^n$ ($(\Diamond^{-1})^n$) makes n R (R^{-1}) steps, i.e.

$$\mathcal{M}, s \models \Diamond^n\varphi \iff \text{there is some } t \in S \text{ such that } sR^nt \text{ and } \mathcal{M}, t \models \varphi$$

and

$$\mathcal{M}, s \models \left(\Diamond^{-1}\right)^n \varphi \iff \text{there is some } t \in S \text{ such that } s\left(R^{-1}\right)^n t \text{ and } \mathcal{M}, t \models \varphi$$

where R^n ($(R^{-1})^n$) denotes the composition of R (R^{-1}) with itself i.e. $R \circ R$ ($R^{-1} \circ R^{-1}$) n times.

The box operators are interpreted in the following way.

$$\mathcal{M}, s \models \Box\varphi \iff \text{for every } t \in S \text{ such that } sRt \text{ it holds that } \mathcal{M}, t \models \varphi$$

$$\mathcal{M}, s \models \Box\varphi \iff \text{for every } t \in S \text{ such that } tRs \text{ it holds that } \mathcal{M}, t \models \varphi$$

2.7 Bisimulation

A bisimulation (bisimulation equivalence) is a relation between two Kripke structures that preserves atomic information and transition possibilities ([83]). Bisimulation will prove its usefulness here when modal formulas expressing reachability need to be model checked.

Definition 2.7.1 ([12]) *Let $\mathcal{F}_1 = (W_1, R_1)$ and $\mathcal{F}_2 = (W_2, R_2)$ be Kripke frames. A **bisimulation (bisimulation equivalence)** between $\mathcal{F}_1$ and $\mathcal{F}_2$ is a non-empty binary relation $B \subseteq W_1 \times W_2$ such that the following conditions hold for B:*

1. *If* $(x, y) \in B$ *then* $\forall x' \in R_1(x)\ \exists y' \in R_2(y)$ *such that* $(x', y') \in B$. *(This is known as the* ***forth condition*** *and expresses that every transition in* $\mathcal{M}_1$ *can be matched by a transition in* $\mathcal{M}_2$.*)*

2. *If* $(x, y) \in B$ *then* $\forall y' \in R_2(y)\ \exists x' \in R_1(x)$ *such that* $(x', y') \in B$. *(This is the* ***back condition*** *and expresses that every transition in* $\mathcal{M}_2$ *can be matched by a transition in* $\mathcal{M}_1$.*)*

If $\mathcal{M}_1 = (\mathcal{F}_1, V_1)$ *and* $\mathcal{M}_2 = (\mathcal{F}_2, V_2)$ *are two Kripke models based on* $\mathcal{F}_1$ *and* $\mathcal{F}_2$ *respectively, then the non-empty binary relation* $B \subseteq W_1 \times W_2$ *is a bisimulation between* $\mathcal{M}_1$ *and* $\mathcal{M}_2$ *if:*

1. B *is a bisimulation from* $\mathcal{F}_1$ *to* $\mathcal{F}_2$, *and*

2. *if* $(x, y) \in B$ *then* $V_1'(x) = V_2'(y)$. *(Related states have identical atomic information)*

Bisimulation relations are used to define bisimilarity between structures.

Definition 2.7.2 ([47]) *Let* $\mathcal{F} = (W, R)$ *be a Kripke frame and let* $w \in W$. *Then the pair* $(\mathcal{F}, w)$ *is a* ***pointed Kripke frame****. Likewise, if* $\mathcal{M}$ *is a Kripke model based on* $\mathcal{F}$, *then the pair* $(\mathcal{M}, w)$ *is a* ***pointed Kripke model****.*

Definition 2.7.3 ([47]) *Let* $\mathcal{G}$ *and* $\mathcal{H}$ *be two Kripke frames (models), with universes* W_1 *and* W_2 *respectively. If* $x \in W_1$ *and* $y \in W_2$, *then the pointed frames (models)* $(\mathcal{G}, x)$ *and* $(\mathcal{H}, y)$ *are* ***bisimilar*** *or* ***bisimulation equivalent****, if there is a bisimulation* B *between* $\mathcal{G}$ *and* $\mathcal{H}$ *such that* B *relates* x *and* y, *i.e.* xBy.

Definition 2.7.4 ([47]) *If* B *is a bisimulation between two Kripke structures* $\mathcal{G}$ *and* $\mathcal{H}$, *such that every element in* $\mathcal{G}$ *is related to some element in* $\mathcal{H}$ *and vice versa, then* B *is a* ***global bisimulation*** *between the two structures, and* $\mathcal{G}$ *and* $\mathcal{H}$ *are said to be* ***globally bisimilar****.*

2.8 Petri nets

In this section a class of models, namely that of Petri nets, is introduced. Petri nets are often used to describe and study systems which are characterised as concurrent or asynchronous. Furthermore, Petri nets admit effective verification tasks. For these and other reasons this class of models have been studied extensively in the literature. In the next chapter it will be explained how the configuration graphs of Petri nets can be seen as rational Kripke frames (rational graphs). However, they are not merely an interesting subclass of rational Kripke frames - Jančar's result on the undecidability of bisimilarity for petri nets will be used to establish a similar result for rational Kripke models.

Definition 2.8.1 ([2, 49, 61, 67]) *A **(labeled) net** is a tuple $(N_l = (P, T, F, L))$ $N = (P, T, F)$ where P and T are finite disjoint sets - the elements of P are called **places** and those of T **transitions**; $F : (P \times T) \cup (T \times P) \to \mathbb{N}$ is called the **flow function** and there is an arc from x to y if $F(x, y) > 0$ with $F(x, y)$ being the multiplicity of that arc. For a labeled net $L : T \to A$ is a labeling that assigns an action name from the finite set A of actions to each transition.*

The places in a net can contain tokens - objects added to a net in order to describe its dynamics. The distribution of these tokens in the places of the net defines its configuration state. For the purpose of this study it can be safely assumed that the tokens do not represent specific information and are indistinguishable.

Definition 2.8.2 ([2, 61, 67]) *A **marking** M is a function $M : P \to \mathbb{N}$ that assigns to each place in a net the number of tokens residing at that place. A marking can also be seen as a finite vector of nonnegative integers where the i^{th} component of M is equal to the number of tokens at the i^{th} place in the net.*

Definition 2.8.3 ([2, 49, 61, 67]) *A **(labeled) Petri net** is a tuple $\mathcal{N} = (N, M_0)$ where N is a (labeled) net and M_0 is the initial marking of the net.*

The distribution of the tokens changes by a process called the firing of transitions. A marking M enables a transition t if $M(p) \geq F(p, t)$ for all $p \in P$ where $F(p, t) > 0$. If the transition t is enabled by the marking M the transition may fire, yielding a new marking M' such that for all $p \in P$ $M'(p) = M(p) - F(p, t) + F(t, p)$. The firing of a transition t is denoted by $M \xrightarrow{t}_N M'$. For labeled Petri nets $M \xrightarrow{a}_N M'$ where $a \in A$ is an action name, denotes that $M \xrightarrow{t}_N M'$ for some transition t such that $L(t) = a$.

Example 2.8.4 Let $N_l = (P, T, F, L)$ be the labeled net such that $P = \{p_1, p_2, p_3, p_4\}$, $T = \{t_1, t_2\}$ and F is given by:

$$F(p_1, t_1) = 3 \quad F(t_1, p_2) = 1 \quad F(t_1, p_3) = 1$$

$$F(p_2, t_2) = 2 \quad F(p_3, t_2) = 1 \quad F(t_2, p_4) = 1$$

and $F(p, t) = 0$ for all other combinations of $t \in T$ and $p \in P$. Finally, if $A = \{a, b\}$, let L be the labeling given by $L(t_1) = a$ and $L(t_2) = b$.

Now, let M_0 be the marking given by (when seen as a finite vector of nonnegative integers) $(3, 1, 0, 0)$.

$\mathcal{N} = (N_l, M_0)$ is a labeled Petri net. The graph in figure 2.3 represents $\mathcal{N}$. The graphs in figures 2.4 and 2.5 represents the Petri net after the first and the second transition are fired respectively. (Places are represented by circles while the transitions are represented by rectangles.)

$M_0(p_1) = 3 = F(p_1, t_1)$ which means that the transition t_1 is enabled by M_0. The marking obtained after t_1 is fired is given by $M_1 = (0, 2, 1, 0)$.

$M_1(p_2) = 2 = F(p_2, t_2)$ and $M_1(p_3) = 1 = F(p_3, t_2)$ which means that M_1 enables t_2. After t_2 is fired the marking $M_3 = (0, 0, 0, 1)$ is obtained.

Figure 2.3: The Petri net $\mathcal{N} = (N_l, M_0)$.

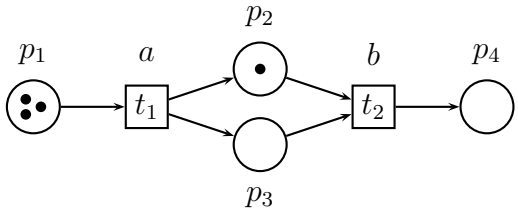


Figure 2.4: $\mathcal{N}$ after transition t_1 is fired.

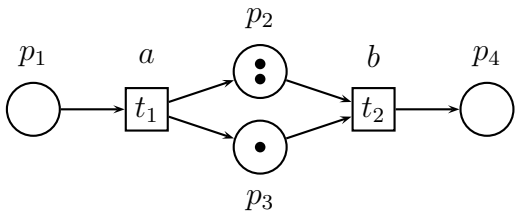
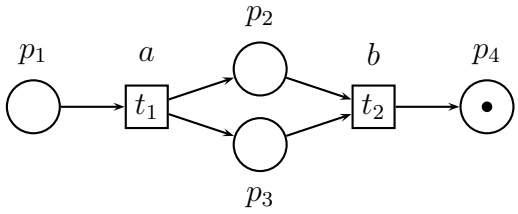


Figure 2.5: $\mathcal{N}$ after transition t_2 is fired.



◁

The markings and firing of transitions can now be used to define the configuration graph of a Petri net.

Definition 2.8.5 ([67]) *The **configuration graph (marking graph)** G of a Petri net $\mathcal{N}$ is the graph $G = (\mathsf{M}, R_t)$ where M is the set containing all valid markings of $\mathcal{N}$ and R a ordered binary relation relating a marking M to a marking M' if and only if there is a transition enabled by M that yields M' when it is fired.*

Chapter 3

Rational Kripke Structures

3.1 Rational graphs

In this section I give a formal definition of rational graphs and present a brief survey of the most important results on this class. This class has been studied by Johnson in [51] and by Morvan in [69, 70].

Definition 3.1.1 *A **graph** is a pair $\mathcal{G} = (S, E)$ where S is the set of vertices of $\mathcal{G}$ and E is a binary relation on S, called the set of edges of $\mathcal{G}$.*

Intuitively rational graphs are defined as below:

Definition 3.1.2 *A graph $\mathcal{G} = (S, E)$ is a **rational graph**, if the set of vertices S is a regular subset of some finitely generated free monoid Σ^* and the set of edges E is a rational relation on Σ^*.*

Labeled rational graphs are simply graphs where there are more than one, but finitely many, edge relations. In this case each edge relation is given a label from some finite alphabet Γ.

From the properties of rational relations (see Chapter 2 on page 14) it then follows that a graph with vertices in Σ^* is rational if and only if its relation is recognised by a finite transducer. In the case of labeled rational graphs, it must be recognised by a labeled finite transducer.

Example 3.1.3 ([69]) The infinite grid is one of the best known rational graphs[1].

Let $\Sigma = \{0, 1\}$, then the infinite grid with vertices in Σ^* is the graph given by figure 3.1. The edge relation of this graph is recognised by the finite transducer given in figure 3.2.

$\lhd$

[1]The transducer given here corrects the transducer given by Morvan

Figure 3.1: The infinite grid with set of vertices $S = 0^*1^*$.

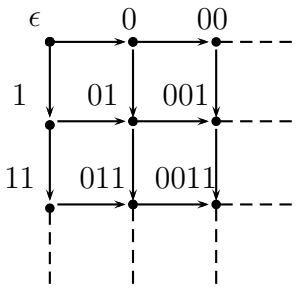
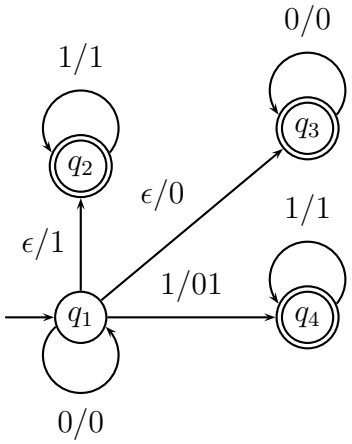


Figure 3.2: A transducer that recognises the infinite grid.



Example 3.1.4 ([77]) Another very important example of a rational graph is the complete binary tree Λ. If $\Sigma = \{0, 1\}$ and $\Gamma = \{a, b\}$, then the graph in figure 3.3 is the complete binary tree with vertices in Σ^* and labeled by Γ, while figure 3.4 is the finite labeled transducer recognising it.

$\triangleleft$

3.1.1 Results on rational graphs

A number of results on rational graphs are given below - some of which are translations of well-known results for rational relations (see Chapter 2).

Proposition 3.1.5 ([69]) *Every rational graph is recursive: it is decidable whether a specific edge belongs to the graph or not.*

Proposition 3.1.6 ([69]) *It is decidable whether or not a rational graph is determin-*

Figure 3.3: The complete binary tree Λ.

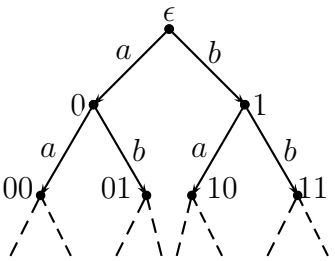
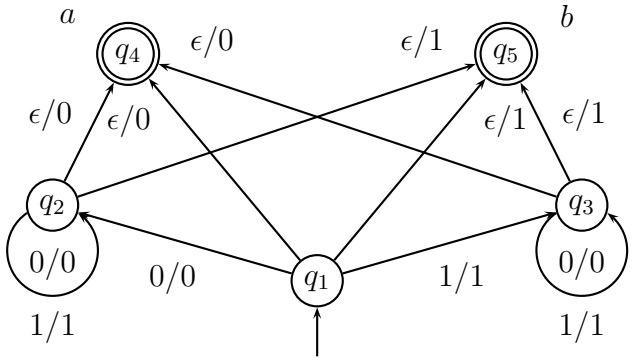


Figure 3.4: A labeled transducer recognising the complete binary tree Λ.



istic - from its transducer.

A rational graph is deterministic if and only if the associated relation on each letter is functional. But functionality is decidable for rational relations.

Proposition 3.1.7 ([69, 7]) *The inclusion and equality of deterministic rational graphs is decidable.*

Consider the following questions:

1. Fix a first-order formula φ. Given a rational graph $\mathcal{G}$ as input, is φ satisfiable on $\mathcal{G}$?

2. Fix a rational graph $\mathcal{G}$. Given a first-order formula φ as input, is φ satisfiable on $\mathcal{G}$?

Negative results have been obtained in answer of both of these questions.

Definition 3.1.8 *Let Σ be a finite alphabet and $(u_1, v_1), (u_2, v_2), \ldots, (u_n, v_n)$ be elements in $\Sigma^* \times \Sigma^*$. Does there exist a sequence of indecis $0 \leq i_1, i_2, \ldots, i_m \leq n$ such that $u_1 u_{i_1} u_{i_2} \cdots u_{i_n} = v_1 v_{i_1} v_{i_2} \cdots v_{i_n}$? This is known as* **Post's Correspondence Problem** *abbreviated by* **PCP**.

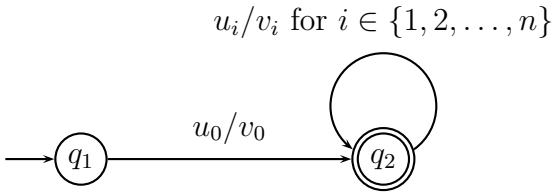
The halting problem for Turing machines is reducible to the PCP and hence the PCP is undecidable.

Theorem 3.1.9 ([69]) *The problem of establishing for a first order formula, given as input a rational graph $\mathcal{G}$ whether or not φ is satisfiable on $\mathcal{G}$ is* **undecidable**.

Proof.[69] In particular Morvan shows that the query $\exists x\,(xRx)$ on rational graphs is undecidable:

Given an instance $\{(u_0, v_0), \ldots, (u_n, v_n)\}$ of the PCP, associate the transducer in figure 3.5 with it.

Figure 3.5: A transducer associated with an instance of the PCP.



Then this instance of the PCP has a solution precisely if a pair (w, w) is accepted by the transducer, i.e. if the first-order formula $\exists x\,(xRx)$ is satisfiable on the graph $\mathcal{G}$ which has the relation recognised by the transducer in figure 3.5. QED

In fact, in [51], Johnson shows that simple first-order definable properties like reflexivity, transitivity and symmetry of a rational relation turn out to be undecidable (with the transducer recognising the relation as input). This was also shown by reduction from the PCP.

Theorem 3.1.10 ([86]) *There is a single rational graph $\mathcal{G}$ for which the problem of establishing whether or not a given first-order formula φ is satisfiable in G is undecidable.*

The details of the proof of this result will be discussed later in this chapter when the model checking of a hybrid extension of the basic tense logic is condidered.

A very important property to verify for structures is reachability. The following result has been obtained with respect to reachability:

Theorem 3.1.11 ([69]) *Reachability (accessibility) is not decidable for rational graphs.*

This problem can also be reduced to the PCP and therefore undecidability follows.

Definition 3.1.12 *[27] The* ***trace*** *of a graph $\mathcal{G}$ labeled by Γ, from a set of initial vertices I to a set of final vertices F is the language $\left\{w \in \Gamma | \exists u \in I, \exists v \in F \left(u \xrightarrow{w}{}^{*} v\right)\right\}$ where $u \xrightarrow{w}{}^{*} v$ denotes that there is a path in $\mathcal{G}$ from vertex u to vertex v such that the concatenation of the labels of the edges in the path forms the word w.*

Proposition 3.1.13 *[27] The traces of rational graphs are exactly the context-sensitive languages.*

This result was extended by Rispal in [78] to a subclass of rational graphs, namely automatic graphs which will be studied in Chapter 4.

Definition 3.1.14 ([66]) *Given a vertex u in a graph $\mathcal{G}$, the* **out-degree** *(respectively,* **in-degree***) of u is the number of different successors (different predecessors) u has. The* **degree** *of u is the sum of its in- and out-degrees. The* **degree of the graph $\mathcal{G}$** *is the largest degree over all its vertices, if it exists, otherwise it is said to be infinite.*

A well-known result for rational relations by Autebert and Boasson in [3] translates into the characterisation of rational graphs with finite out-degree given below.

Proposition 3.1.15 ([69]) *A rational graph $\mathcal{G}$ is of finite out-degree if and only if there exists a transducer $\mathcal{T}$ that recognises the relation of $\mathcal{G}$ such that every cycle in $\mathcal{T}$ labeled with ϵ as input, is labeled with ϵ as output. (i.e. the concatenation of all the input labels on the cycle is just the empty word, then so must the concatenation of all the output labels on the cycle be.)*

For such graphs with finite out-degree Morvan presents the following property on the growth rate of the out-degree of rational graphs:

Theorem 3.1.16 ([69]) *For any rational graph G of finite out-degree and any vertex u there exists $c \in \mathbb{N}$, such that the out-degree of vertices at distance n of u is at most c^{c^n}.*

Example 3.1.17 ([66]) Consider the transducer $\mathcal{T}$ in figure 3.6. The graph recognised by it is presented in figure 3.7. It consists of infinitely many disconnected subgraphs - only two of which are displayed in the figure[2]. The vertices of this graph have out-degree $2^{2^{n+1}}$ at a distance n from any chosen vertex u.

◁

[2]Meyer only displays the first subgraph and makes no reference to the remaining subgraphs.

Figure 3.6: The finite transducer $\mathcal{T}$.

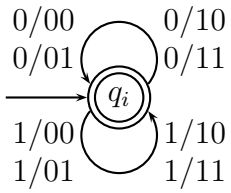
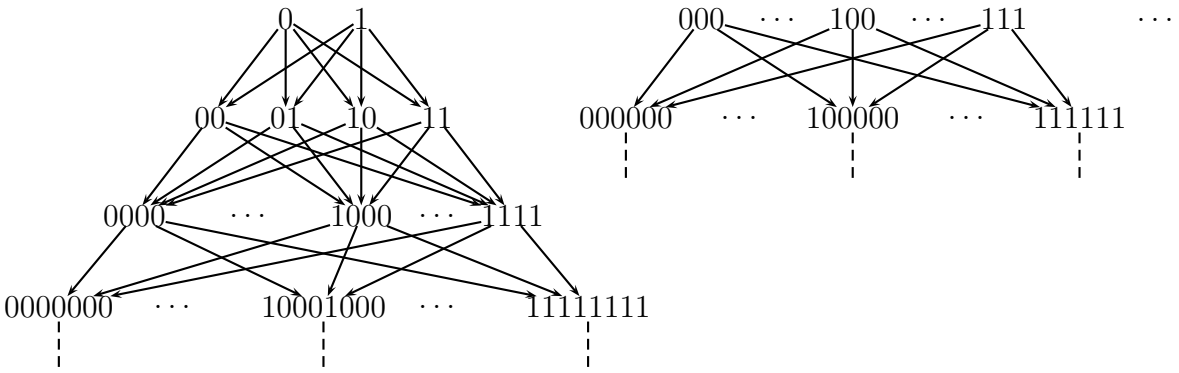


Figure 3.7: A finite degree rational graphs with the vertices at a distance n from any chosen vertex u having out-degree $2^{2^{n+1}}$.



Conversely, Meyer also presents a result for the in-degree:

Theorem 3.1.18 ([66]) *For any rational graph $\mathcal{G}$ of finite in-degree and any vertex u there exists a $c \in \mathbb{N}$ such that c^{c^n} is an upper bound for the in-degree at distance n from u.*

This follows from the fact that the inverse of a rational relation is itself a rational relation.

These properties can be used to show that some classes of graphs are strictly included in the class of rational graphs, or incomparable with the class of rational graphs.

Proposition 3.1.19 ([69]) *The configuration graph of a Petri net is a rational graph.*

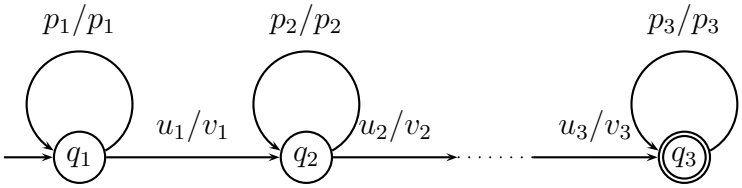
Proof. Let $\mathcal{N} = (P, T, F, M_0)$ be a Petri net with $P = \{p_1, p_2, \ldots, p_n\}$ and $T = \{t_1, t_2, \ldots, t_m\}$ (recall that P and T are both finite by definition). The configuration graph $C = (\mathsf{M}, R)$ can now be obtained as follows: In the alphabet $\{p_1, p_2, \ldots, p_n\}$ a

marking $M = (k_1, \ldots, k_n)$ can be represented by the word $p_1^{k_1} p_2^{k_2} \ldots p_n^{k_n}$. M is then the set of all finite words in the alphabet (note that this set is more than is required) and R is the relation obtained by the union of the m relations associated with the transitions in $\mathcal{N}$. They are of the form: for $1 \leq i \leq m$ let $R_i = \left\{ p_1^{k_1} p_2^{k_2} \ldots p_n^{k_n} \xrightarrow{t_i} p_1^{l_1} p_2^{l_2} \ldots p_n^{l_n} \right\}$ where $\xrightarrow{t_i}$ denotes the firing of the transition t_i.

For each $i \in \{1, 2, \ldots, n\}$ the relation R_i is rational: since F is a function, i.e. $F(p_j, t_i)$ and $F(t_i, p_j)$ have fixed values for each $j \in \{1, 2, \ldots, n\}$, the value $d_{ij} = F(t_i, p_j) - F(p_j, t_i)$ is also fixed. If $d_{ij} > 0$, then let $u_j = \epsilon$ and $v_j = p_j^{d_{ij}}$. Else, if $d_{ij} < 0$ let $u_j = p_j^{|d_{ij}|}$ and $v_j = \epsilon$. Then the relation R_i is recognised by the transducer in figure 3.8. Consequently R_i is a rational relation by Theorem 2.5.2.

But then R is the union of a finite number of rational relations and hence R is itself a rational relation.

Figure 3.8: The transducer recognising the relation R_i associated with the transition t_i.



QED

3.2 Introducing rational Kripke structures

In Chapter 2 Kripke frames and models (collectively referred to as structures) were introduced as the relational structures in which modal languages are interpreted. Rational Kripke structures is a subclass of these structures which are introduced by Goranko in [46].

Definition 3.2.1 *$\mathcal{F} = (S, R)$ is a* **rational Kripke frame** *if there is a finite alphabet Σ such that $S \in \mathsf{REG}(\Sigma^*)$ and the relation R is rational, i.e. $R \in \mathsf{RAT}(\Sigma^* \times \Sigma^*)$.*

From the definition of **rational Kripke frames** it follows that they form exactly the class of **rational graphs**. Hence, all the properties and characterisations of rational graphs carry over to rational Kripke frames.

Definition 3.2.2 *A Kripke model $\mathcal{M} = (\mathcal{F}, V) = (S, R, V)$ is a* **rational Kripke model** *if the frame $\mathcal{F}$, on which it is based, is a rational Kripke frame, and the valuation*

V assigns a regular language to each propositional variable, i.e. $\forall p \in \phi,\ V(p) \in \mathsf{REG}(\Sigma^*)$. *A valuation satisfying this condition is called a* **rational valuation**.

In order for these structures to be useful, they must be effectively finitely presentable.

Proposition 3.2.3 ([46]) *The class of rational Kripke frames (respectively models) are finitely presentable.*

Proof. Let $\mathcal{F} = (\Sigma^*, R)$ be a rational Kripke frame. Since $\Sigma \in \mathsf{REG}(\Sigma^*)$ there is a finite automaton recognising it (see Theorem 2.4.10 on page 13). Furthermore, the rational relation R is recognised by a finite transducers by Theorem 2.5.2. Therefore, $\mathcal{F}$ is effectively finitely presentable.

If $\mathcal{M} = (\mathcal{F}, V)$ now is a rational Kripke model based on $\mathcal{F}$, then all that remains is to show that the valuation V is effectively finitely presentable. The set Φ is finite, and $V(p) \in \mathsf{REG}(\Sigma^*),\ \forall p \in \Phi$. Thus, V can be presented by a finite number of finite automaton, completing the proof. QED

Example 3.2.4 In this example I will present a rational Kripke model based on the configuration graph of a Petri net.

Let $\mathcal{N} = (P, T, F, M_0)$ be a Petri net where $P = \{p_1, p_2\}$, $T = \{t\}$ and $M_0 = (2, 2)$. The flow function is defined by: $F(p_1, t) = 1$, $F(t, p_2) = 2$ and $F(t, p_1) = F(p_2, t) = 0$.

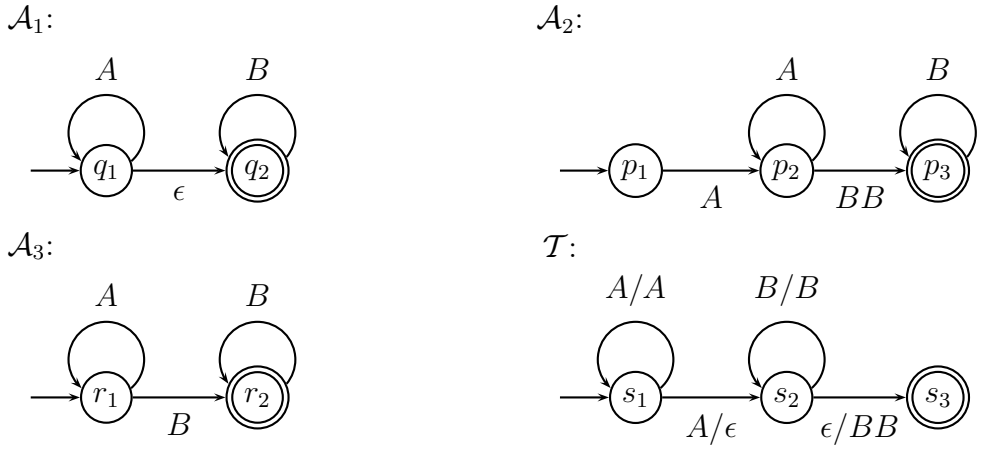
Let $\Sigma = \{A, B\}$ and $\Phi = \{q_1, q_2\}$. Now let $\mathcal{M} = (S, R, V)$ be the Kripke model where $S = A^*B^*$, R is the transition relation of $\mathcal{N}$ and V is the valuation defined by $V(q_1) = A^+BB^+$ and $V(q_2) = A^*B^+$. Then $\mathcal{M}$ is a rational Kripke model which can be presented by the various machines in figure 3.9.

◁

3.3 Model checking

Model checking is verifying properties of a system specified in some formal language in an algorithmic way. Originally model checking was only developed for finite state systems [30, 31] (see also [32] for more on model checking). However, verification problems can be formulated as algorithmic problems for infinite state systems if the systems are finitely presentable - thus making them model checking problems. In the case of finite systems, doing model checking is always decidable. There is however the problem of state space explosion to complicate matters. For infinite state systems, decidability of model checking is not guaranteed. Thus the aim is to identify classes of structures (systems) on which certain model checking problems do turn out to be decidable. The main problem here is that it is always a trade-off between expressiveness of the formal language and decidability of model checking properties specified in the

Figure 3.9: A finite presentation $\mathcal{M}$: $\mathcal{A}_1$, $\mathcal{A}_2$ and $\mathcal{A}_3$ recognises S, $V(q_1)$ and $V(q_2)$ respectively, and $\mathcal{T}$ recognises R.



aforementioned language. For a larger class of structures expressiveness of the language, in which the properties are specified, needs to be sacrificed in order to retain the decidability of the model checking of these properties. On the other hand, insisting on high expressive power will automatically reduce the class of structures on which decidability is guaranteed.

In this section I am particularly interested in decidable model checking tasks on rational Kripke structures. In chapter 4 subclasses of rational Kripke structures, for which model checking problems in more expressive languages are decidable, are investigated.

There are a number of computational tasks, i.e. model checking problems, for modal languages which can be performed on Kripke structures ([46]):

1. Model checking tasks for Kripke models:

   (a) **Local model checking:** Determine whether or not $\mathcal{M}, s \models \varphi$ for $\mathcal{M}$ a Kripke model, s a state in $\mathcal{M}$ and φ a formula.

   (b) **Global model checking:** Determine the set of all states where a formula φ is true in a Kripke model $\mathcal{M}$. Denote the set of all such states by $[\![\varphi]\!]_{\mathcal{M}}$.

   (c) **Satisfiability checking:** Determine whether or not $[\![\varphi]\!]_{\mathcal{M}} = \emptyset$ for a Kripke model $\mathcal{M}$ and a formula φ.

2. Model checking tasks for Kripke frames:

   (a) **Local validity checking:** Determine whether or not $\mathcal{F}, s \models \varphi$ for all possible valuations V on $\mathcal{F}$.

(b) **Global validity checking:** Determine the set of state in a Kripke frame $\mathcal{F}$ at which a formula φ is valid.

(c) **Rational validity checking:** Determine whether or not $\mathcal{F}, s \models \varphi$ for all rational valuations on $\mathcal{F}$.

3.3.1 Synchronised products of transducers and automata

Work in this section, as well as the following four sections, appear in [6] co-authored by myself and Goranko.

The synchronised product of a transducer and an automaton is introduced in this section. This concept will be used when the model checking of K_t-formulas is considered on rational Kripke models. However, some additional terminology must be introduced before the synchronised product of a transducer and an automaton can be defined.

In this section ϵ will denote the empty word, but will also be treated as a special symbol in an extended alphabet.

Definition 3.3.1 ([6]) *Let* $\mathbf{u}$ *be a word in some alphabet* Γ *and* $\gamma \in \Gamma$. *The* γ***-reduction*** *of* $\mathbf{u}$, *denoted* $\mathbf{u}|_\gamma$, *is the word obtained from* $\mathbf{u}$ *after deleting all occurrences of* γ. *Likewise, if* Y *is a language in the alphabet* Γ, *then the* γ***-reduction*** *of* Y, *denoted* $Y|_\gamma$, *is the language consisting of all* γ*-reductions of words in* Y.

Lemma 3.3.2 ([6]) *If* Y *is a regular language over an alphabet* Γ *then* $\mathbf{Y}|_\gamma$ *is a regular language over the alphabet* $\Gamma - \{\gamma\}$.

Proof. An automaton $\mathcal{A}|_\gamma$ recognising $\mathbf{Y}|_\gamma$, called here the γ**-reduction** of $\mathcal{A}$ can be constructed from an automaton $\mathcal{A}$ recognising Y as follows:

1. remove all γ-transitions;
2. add (q, γ', q'') as a transition in $\mathcal{A}|_\gamma$ whenever (q, γ, q') and (q', γ', q'') are transitions in $\mathcal{A}$ and $\gamma \neq \gamma'$;
3. Finally, define the accepting states of $\mathcal{A}|_\gamma$ as all accepting states of $\mathcal{A}$ plus those states q such that $(q \xrightarrow{\gamma^*} q')$ in $\mathcal{A}$ and q' is an accepting state in $\mathcal{A}$.

QED

Example 3.3.3 Let $\Sigma = \{0, 1, 2\}$ and $\mathcal{A}$ the automaton given in figure 3.10. Then the automaton in figure 3.11 is the 2**-reduction** of $\mathcal{A}$, i.e. $\mathcal{A}|_2$. The transitions $(p_4, 2, p_2))$ and $(p_2, 2, p_5)$ are removed from $\mathcal{A}$, the transitions $(p_4, 1, p_4)$ and $(p_2, 0, p_3)$ added and p_4 is made an accepting state.

$\lhd$

Figure 3.10: The automaton $\mathcal{A}$ before reduction

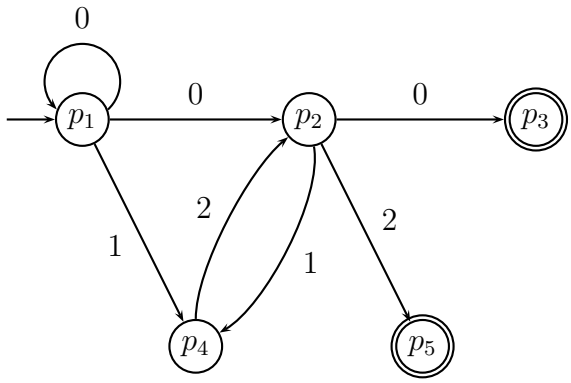
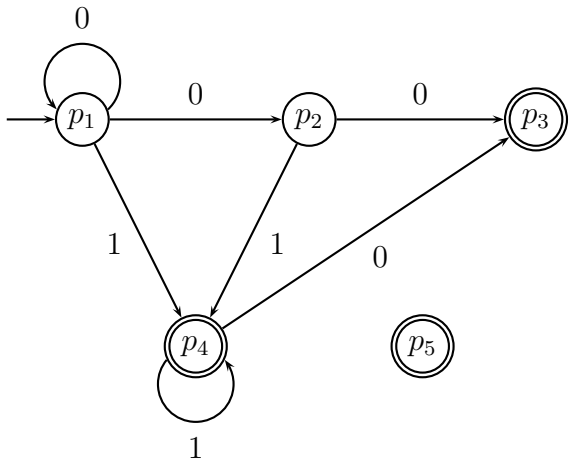


Figure 3.11: The automaton $\mathcal{A}|_2$, the 2-reduction of $\mathcal{A}$



Definition 3.3.4 ([6]) *A **stuttering run** of a finite automaton $\mathcal{A} = \langle Q, \Sigma, q^0, F, \delta \rangle$ is a sequence $q_0 \xrightarrow{x_1} q_1 \xrightarrow{x_2} q_2 \cdots \xrightarrow{x_n} q_n$, such that $q_0 = q^0$, $q_j \in Q$, and either $x_j \in \Sigma$ and $q_j \in \delta(q_{j-1}, x_j)$, or $x_j = \epsilon$ and $q_j = q_{j-1}$ for every $j = 1, 2, \ldots, n$.*

*Thus, a stuttering run of an automaton can be obtained by inserting ϵ-transitions from a state to itself into a run of that automaton. If the latter run is accepting, the stuttering run is said to be an **accepting stuttering run**.*

*A **stuttering word** in an alphabet Σ is any word in $\Sigma \cup \{\epsilon\}$.*

*The **stuttering language** of the automaton $\mathcal{A}$ is the set $L^{\epsilon}(\mathcal{A})$ of all stuttering words whose ϵ-reductions are recognised by $\mathcal{A}$; equivalently, all stuttering words for which there is an accepting stuttering run of the automaton.*

Definition 3.3.5 ([6]) *Let $\mathcal{T} = \langle Q_{\mathcal{T}}, \Sigma, q^0_{\mathcal{T}}, F_{\mathcal{T}}, \rho_{\mathcal{T}}\rangle$ be a transducer (here the input and output alphabets coincide, i.e. $\Gamma = \Sigma$.[3]), and let $\mathcal{A} = \langle Q_{\mathcal{A}}, \Sigma, q^0_{\mathcal{A}}, F_{\mathcal{A}}, \delta_{\mathcal{A}}\rangle$ be a (non-deterministic) finite automaton.*

*The **synchronised product** of $\mathcal{T}$ with $\mathcal{A}$ is the finite automaton:*

$$\mathcal{T} \prec \mathcal{A} = \left\langle Q_{\mathcal{T}} \times Q_{\mathcal{A}}, \Sigma, \left(q^0_{\mathcal{T}}, q^0_{\mathcal{A}}\right), F_{\mathcal{T}} \times F_{\mathcal{A}}, \delta_{\mathcal{T}\prec\mathcal{A}}\right\rangle$$

where $\delta_{\mathcal{T}\prec\mathcal{A}} : (Q_{\mathcal{T}} \times Q_{\mathcal{A}}) \times (\Sigma \cup \{\epsilon\}) \to \mathcal{P}(Q_{\mathcal{T}} \times Q_{\mathcal{A}})$ is such that, for any $p^1_{\mathcal{T}}, p^2_{\mathcal{T}} \in Q_{\mathcal{T}}$ and $p^1_{\mathcal{A}}, p^2_{\mathcal{A}} \in Q_{\mathcal{T}}$ then $(p^2_{\mathcal{T}}, p^2_{\mathcal{A}}) \in \delta_{\mathcal{T}\prec\mathcal{A}}\left((p^1_{\mathcal{T}}, p^1_{\mathcal{A}}), x\right)$ if and only if

1. *either there exists a $y \in \Sigma$ such that $\delta_{\mathcal{A}}(p^1_{\mathcal{A}}, y) = p^2_{\mathcal{A}}$ and $(p^1_{\mathcal{T}}, x, y, p^2_{\mathcal{T}}) \in \rho_{\mathcal{T}}$;*
2. *or $(p^1_{\mathcal{T}}, x, \epsilon, p^2_{\mathcal{T}}) \in \rho_{\mathcal{T}}$ and $p^1_{\mathcal{A}} = p^2_{\mathcal{A}}$.*

Note that every run $R_{\mathcal{T}\prec\mathcal{A}} = (p^0_{\mathcal{T}}, p^0_{\mathcal{A}}) \stackrel{u_1}{\to} (p^1_{\mathcal{T}}, p^1_{\mathcal{A}}) \stackrel{u_2}{\to} \cdots \stackrel{u_n}{\to} (p^n_{\mathcal{T}}, p^n_{\mathcal{T}})$ of the automaton $\mathcal{T} \prec \mathcal{A}$ can be obtained from a pair:
a run $R_{\mathcal{T}} = p^0_{\mathcal{T}} \stackrel{(u_1/w_1)}{\to} p^1_{\mathcal{T}} \stackrel{(u_2/w_2)}{\to} p^2_{\mathcal{T}} \cdots \stackrel{(u_n/w_n)}{\to} p^n_{\mathcal{T}}$ in $\mathcal{T}$,
and a stuttering run $R^s_{\mathcal{A}} = p^0_{\mathcal{A}} \stackrel{w_1}{\to} p^1_{\mathcal{A}} \stackrel{w_2}{\to} p^2_{\mathcal{A}} \cdots \stackrel{w_n}{\to} p^n_{\mathcal{A}}$ in $\mathcal{A}$,
by pairing the respective states $p^j_{\mathcal{T}}$ and $p^j_{\mathcal{A}}$ and removing the output symbol w_j for every $j = 1, 2, \ldots, n$.

Let the reduction of $R^s_{\mathcal{A}}$ be the run $R_{\mathcal{A}} = q^0_{\mathcal{A}} \stackrel{v_1}{\to} q^1_{\mathcal{A}} \stackrel{v_2}{\to} q^2_{\mathcal{A}} \cdots \stackrel{v_m}{\to} q^m_{\mathcal{A}}$, with $m \leq n$. Then the run $R_{\mathcal{T}\prec\mathcal{A}}$ is a **synchronisation of the runs** $R_{\mathcal{T}}$ and $R_{\mathcal{A}}$.

Note, that the synchronisation of accepting runs of $\mathcal{T}$ and $\mathcal{A}$ is an accepting run of $R_{\mathcal{T}\prec\mathcal{A}}$. The following lemma is now immediate:

Lemma 3.3.6 ([6]) *Let $\mathcal{T} = \langle Q_{\mathcal{T}}, \Sigma, q^0_{\mathcal{T}}, F_{\mathcal{T}}, \rho_{\mathcal{T}}\rangle$ be a transducer recognising the relation $R(\mathcal{T})$ and let $\mathcal{A} = \langle Q_{\mathcal{A}}, \Sigma, q^0_{\mathcal{A}}, F_{\mathcal{A}}, \delta_{\mathcal{A}}\rangle$ be a finite automaton recognising the language $L(\mathcal{A})$. Then the language recognised by the synchronised product of $\mathcal{T}$ and $\mathcal{A}$ is*

$$L(\mathcal{T} \prec \mathcal{A}) = \{\mathbf{u} \mid \exists \mathbf{w} \in L^{\epsilon}(\mathcal{A})(\mathbf{u}R(\mathcal{T})\mathbf{w})\}.$$

Example 3.3.7 Consider the transducer $\mathcal{T}$ and the automaton $\mathcal{A}$ given in figures 3.12 and 3.13 respectively. The regular language $X = 1^*(1 + 0^+)$ is recognised by $\mathcal{A}$, while

$$R = \left\{(1^n 0, 10^n 1)^m \left(1^k, 10^k\right) | n, m, k \in \mathbb{N}\right\} \cup \left\{(1^n 0, 10^n 1)^m \left(01^k, 11^k\right) | n, m, k \in \mathbb{N}\right\}$$

is the relation recognised by $\mathcal{T}$. Then their synchronised product $\mathcal{T} \prec \mathcal{A}$ is the finite automaton given in figure 3.14. (This automaton can be simplified by removing redundant states and edges.)

$\lhd$

[3]Henceforth $\Gamma = \Sigma$, i.e. the input and output alphabets of transducers considered coincide - unless explicitly specified otherwise

Figure 3.12: The transducer $\mathcal{T}$ used in the synchronised product.

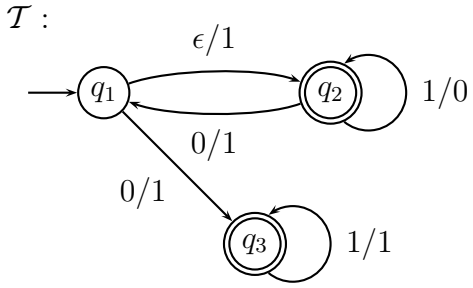


Figure 3.13: The automaton $\mathcal{A}$ used in the synchronised product.

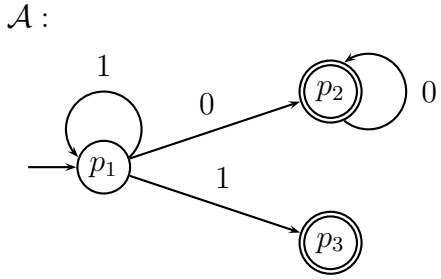
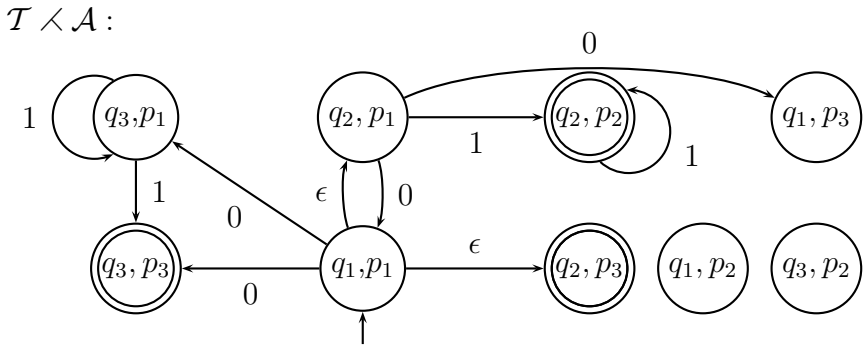


Figure 3.14: The synchronied product $\mathcal{T} \prec \mathcal{A}$.



3.3.2 Model checking of K_t in rational Kripke models

Finally, the three model checking tasks for formulas from K_t can be considered in rational Kripke models.

Lemma 3.3.8 ([6]) *Let Σ be a finite non-empty alphabet, $X \subseteq \Sigma^*$ a regular subset, and let $R \subseteq \Sigma^* \times \Sigma^*$ be a rational relation. Then the sets*

$$\langle R \rangle X = \{u \in \Sigma^* | \exists v \in X(uRv)\}$$

and

$$\left\langle R^{-1} \right\rangle X = \{u \in \Sigma^* | \exists v \in X(vRu)\}$$

are regular subsets of Σ^*.

Proof. This claim essentially follows from Nivat's theorem (see Theorem 2.5.4). However, using lemmas 3.3.2 and 3.3.6 a constructive proof is given in which automata that recognise the resulting regular languages are explicitly produced: for $\langle R \rangle X$ that is the ϵ-reduction of the synchronised product of $\mathcal{T}$ with $\mathcal{A}$; for $\langle R^{-1} \rangle X$ the transducer for R^{-1} is used instead of $\mathcal{T}$. This transducer is obtained from $\mathcal{T}$ by swapping the input and output symbols in the transition relation.

Note that, in general the resulting automata may have redundant states and transitions. QED

Example 3.3.9 Recall the synchronised product of the automaton recognising the regular language $X = 1^* (1 + 0^+)$ and the transducer recognising the rational relation:

$$R = \left\{ (1^n 0, 10^n 1)^m \left(1^k, 10^k\right) | n, m, k \in \mathbb{N} \right\} \cup \left\{ (1^n 0, 10^n 1)^m \left(01^k, 11^k\right) | n, m, k \in \mathbb{N} \right\},$$

given in figure 3.14, example 3.3.7. From lemma 3.3.8 it now follows that this synchronised product recognises exactly the regular language $\langle R \rangle X = 0^* + 0^* 1^+$. $\lhd$

Theorem 3.3.10 ([46]) *Local and global model checking of formulas in K_t in rational Kripke models are decidable. Moreover, for every formula $\varphi \in K_t$ and rational Kripke model $\mathcal{M}$, the set $[\![\varphi]\!]_{\mathcal{M}}$ is an effectively computable rational language.*

Proof. Let $\mathcal{M} = (\Sigma^*, R, V)$ be a rational Kripke model over a finite alphabet Σ and $\varphi \in K_t$.

The set $[\![\varphi]\!]_{\mathcal{M}}$ is a regular language, by induction on φ:

1. When φ is an atomic proposition, the claim follows from the definition of a rational model.

2. The boolean cases follow from the closure of regular languages under boolean operations.

3. If $\varphi = \langle R \rangle \psi$ then $[\![\varphi]\!]_{\mathcal{M}} = \langle R \rangle [\![\psi]\!]_{\mathcal{M}}$, which is regular by the inductive hypothesis and lemma 3.3.8. Likewise for the case $\varphi = \langle R^{-1} \rangle \chi$.

Finally, decidability of the local model checking follows from the decidability of the non-emptiness of a regular language.

QED

3.3.3 Complexity

The complexity of global model checking a formula in K_t on a given rational Kripke model is now analysed, i.e., the **expression complexity** [58] of K_t on rational Kripke models.

A formula $\varphi \in K_t$ is in **negation normal form** if every occurrence of the negation immediately precedes a propositional variable. Clearly every formula $\varphi \in K_t$ is equivalent to a formula $\psi \in K_t$ in negation normal form, of size linear in the size φ. For the remainder of this section, a formula φ which is to be model checked is assumed to be in a negation normal form.

The number of edges in a finite automaton or transducer $\mathcal{M}$ is denoted by $|\mathcal{M}|$.

Lemma 3.3.11 ([6]) *If $\mathcal{A}$ is an automaton recognising the regular language X and $\mathcal{T}$ a transducer recognising the rational relation R, then $O(|\mathcal{T}|^m|\mathcal{A}|)$ is an upper bound for the time complexity of computing an automaton recognising $\langle R\rangle^m X$.*

Proof. The size of the synchronised product $\mathcal{T} \curlywedge \mathcal{A}$ of $\mathcal{T}$ and $\mathcal{A}$ is bounded above by $|\mathcal{T}||\mathcal{A}|$ and it can be computed in time $O(|\mathcal{T}||\mathcal{A}|)$. The claim now follows by iterating that procedure m times. QED

However, the conjecture is that the time complexity of computing an automaton recognising $[R]\, X$ is far worse.

For a regular language X recognised by an automaton $\mathcal{A}$, define $R_X = \{(u, \epsilon)\,|u \in X\}$. A transducer $\mathcal{T}$ recognising R_X can be constructed from $\mathcal{A}$ by simply replacing every edge (q, x, p) in $\mathcal{A}$ with the edge (q, x, ϵ, p).

Lemma 3.3.12 ([6]) *Let X be a regular language. Then the complementation $\overline{X}$ of X equals $[R_X]\perp$.*

Proof.

$$[R_X]\perp = \{u \in \Sigma^* | R_X(u) \subseteq \emptyset\}$$

Thus, $[R_X]\perp$ will contain only those words in Σ^* with no R_X successors, i.e. exactly the set $\overline{X}$. QED

Consequently, computing $[R_X]\perp$ cannot be done in less than exponential time in the size of the (non-deterministic) automaton $\mathcal{A}$ for X. This result suggests the following conjecture:

Definition 3.3.13 ([6]) *The* **alternating box rank** *and* **alternating diamond rank** *of a formula $\varphi \in K_t$, denoted respectively by $ar_{[R]}(\varphi)$ and $ar_{\langle R\rangle}(\varphi)$, are defined by simultaneous induction as follows, where $\triangle \in \{[R], \langle R\rangle\}$:*

1. *if p is an atomic proposition, then $ar_{\triangle}(p) = 0$;*

2. $ar_{[R]}(\neg\psi) = ar_{\langle R\rangle}(\psi)$ *and* $ar_{\langle R\rangle}(\neg\psi) = ar_{[R]}(\psi)$;

3. $ar_{\triangle}(\psi_1 \vee \psi_2) = \max\{ar_{\triangle}(\psi_1), ar_{\triangle}(\psi_2)\}$;

4. $ar_{\langle R\rangle}(\langle R\rangle\psi) = ar_{[R]}(\psi) + 1$ *and* $ar_{[R]}(\langle R\rangle\psi) = ar_{[R]}(\psi)$.

5. $ar_{[R]}([R]\psi) = ar_{\langle R\rangle}(\psi) + 1$ *and* $ar_{\langle R\rangle}([R]\psi) = ar_{\langle R\rangle}(\psi)$.

The alternating box (diamond) rank of a formula counts the greatest number of nested alternations of modalities with an outmost box (diamond) in the formula.

Conjecture 3.3.14 ([6]) *The expression complexity of global model checking of a K_t-formula is non-elementary in terms of the alternating box rank of the formula.*

3.3.4 Hybrid logics

Hybrid languages

One of the main shortcomings of the K_t is its inability to refer to the states of a Kripke structure. Modal semantics evaluate modal formulas at some state in the model, but using standard modal languages, the states themselves cannot be referred to.

A remedy to this problem was introduced in the form of hybrid logics. For more on hybrid logics see [11]. In its basic form, a hybrid language is an extention of K or K_t with a new set of symbols which enables one to refer to specific states in a model.

Definition 3.3.15 ([11]) *Let Φ be a set of propositional variables and let Θ be a set of symbols called* **nominals**, *such that $\Theta \cap \Phi = \emptyset$. A formula φ is in the* **basic hybrid language** *H_t if it satisfies the rule*

$$\varphi = p \mid i \mid \neg\varphi \mid \varphi \vee \phi \mid \Diamond\varphi \mid \Diamond^{-1}\varphi$$

where $\Diamond$ is the unary diamond operator in modal languages, $p \in \Phi$ and $i \in \Theta$.

Hence, the only difference between K_t and H_t is the addition of a second type of atomic formula, namely the nominals. The Kripke frames on which hybrid languages are interpreted, called **hybrid frames**, are exactly the same as the frames on which K_t is interpreted. On the level of models the valuation must be altered slightly: let $\mathcal{M} = (\mathcal{F}, V)$ be a **hybrid model** based on the hybrid frame $\mathcal{F}$, then V is the function $V : \Phi \cup \Theta \to \mathcal{P}(S)$ where S is the domain of the model.

A nominal must be true at exactly one state in the model, i.e. $\mathbf{V}(i)$ **is a singleton** for every nominal i in Θ. The unique state in $V(i)$ is called the **denotation** of i. Therefore, if $\mathcal{M} = (S, R, V)$ is a hybrid model and $w \in S$, then the formula $\varphi = i$ for $i \in \Theta$ is satisfied at w in $\mathcal{M}$, i.e. $\mathcal{M}, w \models i$ if and only if $V(i) = \{w\}$ (w is the denotation of i).

Example 3.3.16 Consider the following formula: $\Diamond\mathbf{i}$. This formula simply states that the denotation of the nominal i is one of the successors. It will be satisfied at all predecessors of the denotation of i. This simple formula expresses a property which cannot be expressed in the basic modal language since there is no way to refer to specific states. $\lhd$

The basic hybrid language can be extended in a number of ways.

Firstly, introduce a unary operator for each nominal i in Θ. This makes it possible to evaluate formulas at a specific point.

Definition 3.3.17 ([11]) *A formula φ is in the hybrid language $H_t(@)$ extended with unary operators - one for each nominal - if it satisfies the rule:*

$$\varphi = p \mid i \mid \neg\varphi \mid \varphi \vee \phi \mid \Diamond\varphi \mid \Diamond^{-1}\varphi \mid @_i\varphi$$

with $i \in \Theta$. The '$@_i$' operators are called the **satisfaction operators**.

The new formula $@_i\varphi$ is read 'φ is true at i'. For the model $\mathcal{M}$ (as used earlier in this section), the formula $\phi = @_i\varphi$ is satisfied at w in $\mathcal{M}$, i.e. $\mathcal{M}, w \models \phi$ if and only if $\mathcal{M}, w' \models \varphi$ where w' is the denotation of i.

Example 3.3.18 The formula $\varphi = @_{\mathbf{i}}(\Diamond\mathbf{j} \wedge \mathbf{p})$ with $i, j \in \Theta$ and $p \in \Phi$ is another example of a very simple formula showing the increase of expressiveness.

Let $w \in S$, then $\mathcal{M}, w \models @_i(\Diamond j \wedge p)$ if and only if for $V(i) = \{w'\}$ it holds that $\mathcal{M}, w' \models \Diamond j \wedge p$. This formula expresses that the state w' has the denotation of j as a successor and $w' \in V(p)$. Note that the state at which the evaluation of φ is done, does not play a role in the satisfaction. Therefore, formulas of the form $@_i\psi$ are either valid or invalid depending on wheter the formula ψ is satisfied at the denotation of i.

Next consider the formula:

$$@_{\mathbf{i}}\phi \leftrightarrow \neg @_{\mathbf{i}} \neg\varphi$$

This formula is valid on all frames and shows that each satisfaction operator is its own dual.

Furthermore, the following formula is valid on all hybrid frames for any nominal i and shows that the satisfaction operators act like normal modal operators.

$$@_{\mathbf{i}}(\varphi \rightarrow \psi) \rightarrow (@_{\mathbf{i}}\varphi \rightarrow @_{\mathbf{i}}\psi)$$

$\lhd$

An even more expressive extension of H_t is obtained when the **universal modality** [**U**] is added to the language.

Definition 3.3.19 *A formula φ is in $H_t(U)$ if it satisfies the rule:*

$$\varphi = p \mid i \mid \neg\varphi \mid \varphi \vee \phi \mid \Diamond\varphi \mid \Diamond^{-1}\varphi \mid [U]\,\varphi$$

Semantically: Let $\mathcal{M} = (S, R, V)$ be a hybrid model, then $\mathcal{M}, v \models [U]\,\varphi \iff \mathcal{M}, w \models \varphi$ for every $w \in S$.

Furthermore, the satisfaction operators $@_i$ are definable in $H_t(U)$: $@_i\varphi := [U]\,(i \to \varphi)$.

Hybrid languages provide a way to access states which naturally leads to a possible further extension: introduce state variables and then bind these variables to states.

Definition 3.3.20 *Let $H_t(U, \downarrow)$ be the extension of $H_t(U)$ with the* **binder**. *It also contains a new type of atomic formula, namely a set of* **state variables** Ψ. *A formula φ is in $H_t(U, \downarrow)$ if it satisfies the rule:*

$$\varphi = p \mid i \mid \neg\varphi \mid \varphi \vee \phi \mid \Diamond\varphi \mid \Diamond^{-1}\varphi \mid [U]\,\varphi \mid \downarrow x.\varphi$$

Satisfaction of formulas in this form is a bit more involved. Let $\mathcal{M}$ be a hybrid Kripke model. For a formula φ containing only unbounded occurrences of a state variable x, where w is a state in the model, let $\varphi\,[x \leftarrow i_w]$ denote substituting x with the nominal i_w in φ where i_w is such that w is its denotation. Then the satisfaction of these formulas is defined by:

$$\mathcal{M}, w \models \downarrow x.\varphi \quad \iff \quad \mathcal{M}, w \models \varphi\,[x \leftarrow i_w]$$

$H_t(U, \downarrow)$ is a lot more expressive than $H_t(U)$.

Example 3.3.21

$$\downarrow \mathbf{x}.@_{\mathbf{i}}\Diamond\mathbf{x}$$

This formula is true at the set of states which are successors of the denotation of the nominal i. ◁

Model checking of hybrid languages on rational Kripke models

Having introduced hybrid languages, an obvious question to ask is whether or not model checking of hybrid logic formulas is decidable. First observe that, as in the case of propositional variables, it is safe to assume that the set of nominals Θ, added to the language $H_t(U)$, is finite, since every formula must have finite length and can therefore only contain finitely many nominals.

Proposition 3.3.22 ([6]) *Global and local model checking the formulas of the hybrid language $H_t(U)$ is decidable on rational Kripke models. As with the basic modal language the set $[\![\varphi]\!]_{\mathcal{M}}$ is effectively computable for every formula φ and every rational Kripke model $\mathcal{M} = (S, R, V)$.*

Proof. This language contains two new types of formulas: the atomic formulas i with $i \in \Theta$ and formulas of the form $[U]\varphi$ where φ is some hybrid formula. For formulas in $H_t(U)$ also contained in K_t the claim follows from Theorem 3.3.10.

The set $V(i)$ for $i \in \Theta$ is a singleton by the definition of nominals. Therefore, $\|i\|_{\mathcal{M}}$ is a regular set for every atomic formula i.

Next consider a formula of the form $[U]\varphi$ and suppose $\|\varphi\|_{\mathcal{M}}$ is a regular set. If $\|\varphi\|_{\mathcal{M}} = S$ then $\|[U]\varphi\|_{\mathcal{M}} = S$, otherwise $\|[U]\varphi\|_{\mathcal{M}} = \emptyset$. Either of these are regular sets and the result follows. QED

Proposition 3.3.23 ([6]) *Model checking the* $\mathbf{H}_t(U,\downarrow)$*-formula* $\downarrow x.\langle R\rangle x$ *on a given input rational Kripke model is not decidable.*

Proof. This is an immediate consequence from the earlier mentioned Morvan's reduction [69] in theorem 3.1.9 of the model checking of $\exists xRxx$ to the Post Correspondence Problem. QED

Proposition 3.3.24 ([6]) *There is a rational Kripke model on which model checking formulas from the hybrid language* $H_t(U,\downarrow)$ *is undecidable.*

Proof. I now present the proof by Thomas [87] where he shows that there is a single rational graph with an undecidable first-order theory. I will then show that the condition under which the instance of the PCP has a solution, can not only be expressed by a first-order sentence, but also by a formula in $H_t(U,\downarrow)$.

Thomas uses a universal Turing machine M and the encoding of its undecidable halting problem (for different input words u) into a family of instances of the PCP:

A Turing machine M with input word u is converted into a PCP instance

$$\{(u_0, v_0), \ldots, (u_m, v, m)\}$$

over an alphabet Σ, such that Σ contains the state and tape letter of M as well as the symbol # used to separate between M-configurations in M-computations. If the input word is $u = a_1 \cdots a_n$ then u_0 is the initial configuration word $c(u) = \#q_0a_1 \cdots a_n$ of M. Furthermore, it is always the case that $v_0 = \#$ and $u_1, \ldots, u_m, v_1, \ldots, v_m$ only depend on M.

Then M halts on the input word u if and only if the PCP instance

$$\{(c(U), \#), (u_1, v_1), \ldots (u_m, v_m)\}$$

has a solution.

Now let $\mathcal{G}$ be the rational graph defined from these PCP instances as in the proof of Theorem 3.1.9: The vertices are the words over Σ, and there is a single edge relation R such that $(w_1, w_2) \in R$ if and only if there are indices $i_1, \ldots, i_l$ such that $w_1 =$

$c(u)\, u_{i_1} \cdots u_{i_l}$ and $w_2 = \#v_{i_1} \cdots v_{i_l}$. The graph $\mathcal{G}$ is rational and has an edge form a word to itself if the instance of the PCP has a solution.

To be able to refer to the input word u explicitly in the graph, further vertices and edge relations R_x for $x \in \Sigma$ are added. A $c(u)$-labeled path via the new vertices will lead to a vertex of $\mathcal{G}$ with prefix $c(u)$. If the latter vertex has an edge back to itself a solution for the PCP instance $\{(c(U), \#), (u_1, v_1), \ldots (u_m, v_m)\}$ can be inferred.

The new vertices are words over a copy $\underline{\Sigma}$ of Σ. The alphabet $\underline{\Sigma}$ consists of underlined versions of the letters in Σ. For any word $c(u)$ vertices which arise from the underlined versions of the proper prefixes of $c(u)$ are added. In addition a R_x-edge from an underlined word w to a word $w\underline{x}$ is introduced (also for the case where $w = \epsilon$). There are also edges to non-underlined words: there is a R_x-edge from the underlined version of w to any non-underlined word which has wx as a prefix. The resulting graph is now called $\mathcal{G}'$.

By construction of $\mathcal{G}'$, the PCP instance $\{(c(U), \#), (u_1, v_1), \ldots (u_m, v_m)\}$ has a solution if and only if there is a path in $\mathcal{G}'$, labeled with the word $c(u)$, from the vertex ϵ to a vertex which has an edge back to itself.

This conditions can be expressed by a first-order sentence. It can also be expressed by a formula in $H_t(U, \downarrow)$:

The vertex ϵ is definable as the only one with outgoing R_x edges but without an ingoing edge of one of the relations R_x:

$$\downarrow x. \left(\bigwedge_{i=1}^{n} \neg \left\langle R_{x_i}^{-1} \right\rangle x \right) \wedge \left(\bigvee_{i=1}^{n} \left\langle R_{x_i} \right\rangle x \right)$$

A path labeled with the word $c(u)$ to a vertex which has an edge back to itself can then be expressed by:

$$\left\langle R_{a_1} \right\rangle \left\langle R_{a_2} \right\rangle \cdots \left\langle R_{a_n} \right\rangle \downarrow y. \langle R \rangle\, y.$$

The claim then follows. QED

3.4 Regularity preserving relations

The fact that the set $\langle R \rangle X$ is regular for a rational relation R and a regular set X, sparks the interest to find out what relations satisfy this property, to study them and to explore some of their properties. This leads to the definition below.

Definition 3.4.1 *A relation $R \subset \Sigma^* \times \Sigma^*$ is* **regularity preserving** *if for every regular language $X \in \mathsf{REG}(\Sigma^*)$, $\langle R \rangle X$ is a regular language.*

Proposition 3.4.2 *A relation R is regularity preserving if and only if the set $[R] X$ is regular for every $X \in \mathsf{REG}(\Sigma^*)$.*

Proof. This follows from the fact that $[R]$ is the dual of $\langle R\rangle$, i.e. $[R]\,\varphi = \neg\,\langle R\rangle\,\neg\varphi$. If $X \in \mathsf{REG}\,(\Sigma^*)$ then $\overline{X} \in \mathsf{REG}\,(\Sigma^*)$ and

$$\begin{array}{rcl} \langle R\rangle\, X \in \mathsf{REG}\,(\Sigma^*) & \iff & \langle R\rangle\, \overline{X} \in \mathsf{REG}\,(\Sigma^*) \\ & \iff & \overline{\langle R\rangle\, \overline{X}} = [R]\, X \in \mathsf{REG}\,(\Sigma^*) \end{array}$$

QED

Furthermore, for a relation R and a set X it holds that:

$$\begin{array}{rcl} \langle R\rangle\, X & = & \{u \in \Sigma^* | \exists v \in X\,(uRv)\} \\ & = & R^{-1}\,(X) \end{array}$$

Since $\langle R\rangle\, X$ is a regular language for all $X \in \mathsf{REG}\,(\Sigma^*)$ if R is a rational relation, it follows that every rational relation is a regularity preserving relation.

Question 3.4.3 *Is every regularity preserving relation a rational relation?*

Implicitly, the question asked is whether or not there are frames outside of the class of rational Kripke frames for which the model checking of K_t is decidable?

The answer to this question is no - there is at least one regularity preserving relation which is not rational.

Proposition 3.4.4 *Let* $R = \{(x^n, y^n x^n)\,|n \geq 0\}$. *Then* $\langle R\rangle\, X = R^{-1}\,(X)$ *for any* $X \in \mathsf{REG}\,(\Sigma^*)$, *is a regular language.*

Proof. Firstly observe that $R^{-1}\,(X) = R^{-1}\,(X \cap \{y^n x^n | n \geq 0\})$. A well-known closure property of context-free languages is that the intersection of a context-free language with a regular language is a context-free language [84]. Thus, to prove this claim, I will determine the form of the context-free language $\mathcal{L} = X \cap \{y^n x^n | n \geq 0\}$ and then show that the inverse image of $\mathcal{L}$ is a regular language.

Let $\mathcal{M} = \langle Q_1, \Sigma, \Gamma, q^0, F_1, \delta_1\rangle$ be a pushdown automaton recognising $\{y^n x^n | n \geq 0\}$, such that:

$$Q_1 = \left\{q^0, q_1\right\} \quad \Sigma = \{x, y\} \quad \Gamma = \{Y\} \quad F_1 = \left\{q^0, q_1\right\}$$

and

$$\begin{array}{c} \delta_1\left(q^0, y, \lambda\right) = \left\{\left[q^0, Y\right]\right\} \\ \delta_1\left(q^0, x, Y\right) = \{[q_1, \lambda]\} \\ \delta_1\left(q_1, x, Y\right) = \{[q_1, \lambda]\} \end{array}$$

Furthermore, let $\mathcal{A} = \langle Q_2, \Sigma, p^0, F_2, \delta_2\rangle$ be a finite deterministic automaton that recognises the language X. Using the proof given in [84] that the intersection of a context-free and regular language is context-free, a pushdown automaton can be constructed which

recognises the language $\mathcal{L}$: Let $\mathcal{N} = \langle Q, \Sigma, \Gamma, (q^0, p^0), F, \delta \rangle$ be the pushdown automaton where $Q = Q_1 \times Q_2$, $F = F_1 \times F_2$ and $\forall p \in Q_2$

$$\delta\left(\left(q^0, p\right), y, \lambda\right) = \left\{\left[\left(q^0, p'\right), Y\right] | \delta_2\left(p, y\right) = p'\right\}$$
$$\delta\left(\left(q^0, p\right), x, Y\right) = \left\{\left[\left(q_1, p'\right), \lambda\right] | \delta_2\left(p, x\right) = p'\right\}$$
$$\delta\left(\left(q_1, p\right), x, Y\right) = \left\{\left[\left(q_1, p'\right), \lambda\right] | \delta_2\left(p, x\right) = p'\right\}$$

From every state in $\mathcal{N}$ where q^0 is the first component, there are two outgoing edges labeled with x and y respectively. From each state in $\mathcal{N}$ with q_1 as the first component there is exactly one outgoing edge labeled with an x. This follows from the determinism of the automaton $\mathcal{A}$. If an edge labeled with x or y is followed from one state to another, respectively x or y is read.

Clearly all computations must start from the state (q^0, p^0). Starting at the state (q^0, p^0) a path can be followed along which every state has q^0 as first component, the states passed are pairwise distinct and the path is labeled by a string of y's. Let the maximum length of such a path be k_1 where $0 \leq k_1 \leq |Q_2|$. Label the states visited by this path of length k_1 with A_0 to A_{k_1} according to the number of y's read to reach the state.

Now start at the state (q^0, p^0) and follow any path p labeled by a string of y's. Let i be the length of this path.

1. For each i, $0 \leq i \leq k_1$, the singleton $\{y^i x^i\}$ is accepted by $\mathcal{N}$, if and only if the state (q_1, p), the state reached after i y's and then i x's are read, belongs to F. (It will always be possible to read i x's form the determinism of the automaton.) Observe that $i = 0$ is the only case where $q = q^0$, and in fact then $p = p_0$. For all other i, $q = q_1$.

2. If $i > k_1$ let A_{c_1} be the label of the state reached after $k_1 + 1$ y's are read. Observe that $0 \leq c_1 \leq k_1$. This will require that p ends at one of the states numbered between c_1 and k_1. Suppose the state labeled by A_l is reached, clearly then l y's were read, after which some multiple of the cycle was read, where $m_1 = k_1 + 1 - c_1$ is the length of the cycle.

   Let (q^0, p) be the state labeled by A_l. If x is read i times it means that the unique edge from (q^0, p) to some state (q_1, p') must be followed after which, as before, a finite number of distinct states can be reached until a cycle must be followed. Say k_2 distinct states can be reached from (q^0, p) (this includes (q_1, p')), then label each of these states with B_j for $1 \leq j \leq k_2$. Let c_2 be the number such that after reading $k_2 + 1$ x's from (q^0, p), the state labeled by B_{c_2} is reached. Thus the cycle reached will have length $m_2 = k_2 + 1 - c_2$.

   Let $t = lcm\left(m_1, m_2\right)$ then the length of the path p is $i = n + st$ for $n = 0, 1, \ldots, t - 1$ and $s \geq 0$ from number theory. This means that there are $t - 1$ languages of the form $\left\{y^n \left(y^t\right)^s x^n \left(x^t\right)^s | s \geq 0\right\}$ that may be accepted by this pushdown automaton.

Thus, the context-free language consists of the union of a finite number of sets of the form $\{y^n (y^t)^s x^n (x^t)^s | s \geq 0\}$ and singletons. The inverse image of a singleton is clearly a singleton while the inverse image of a set $\{y^n (y^t)^s x^n (x^t)^s | s \geq 0\}$ is the set $\{x^n (x^t)^s | s \geq 0\}$ which is a regular language. Thus, the inverse image of the language $\mathcal{L}$ is the union of a finite number of regular languages - which is regular itself.

QED

Example 3.4.5 Let $X = y (y^3)^* x (x^2)^*$. For the pushdown automaton constructed using a deterministic automaton for X, it follows that $k_1 = 3$, $c_1 = 1$ and $m_1 = 3$.

Then, in the case where $i < k_1$, only the $\{xy\}$ is accepted by this pushdown automaton.

Furthermore, $k_2 = 2$, $c_2 = 1$ and $m_2 = 2$. This means that $t = lcm(m_1, m_2) = 6$ and that the languages $\left\{y^n (y^6)^s x^n (x^6)^s | s \geq 0\right\}$ for $n = 1, 2, \ldots, 5$ are the languages that may possibly be accepted by the pushdown automaton. Of these only the language $\left\{y (y^6)^s x (x^6)^s | s \geq 0\right\}$ is accepted. The inverse image of this language is the language given by $\left\{x (x^6)^s | s \geq 0\right\} \cup \{x\}$ which is a regular language. ◁

The next point of interest is the closure properties of this class of structures. The next theorem presents some positive results.

Theorem 3.4.6 *The class of regularity preserving relations is* **closed** *under* **union** *and* **composition**.

Proof. Let R_1 and R_2 be two regularity preserving relations, i.e. for every $X \in \mathsf{REG}(\Sigma^*)$ the languages $[R_1] X$ and $[R_2] X$ are regular.

1. Union:

$$\begin{aligned} [R_1 \cup R_2] X &= \{u \in \Sigma^* | (R_1 \cup R_2)(u) \subseteq X\} \\ &= \{u \in \Sigma^* | (R_1(u) \cup R_2(u)) \subseteq X\} \\ &= \{u \in \Sigma^* | R_1(u) \subseteq X \wedge R_2(u) \subseteq X\} \\ &= [R_1] X \cap [R_2] X \end{aligned}$$

   This is the intersection of two regular languages, which is itself a regular language.

2. Composition:

$$\begin{aligned} [R_1 \circ R_2] X &= \{u \in \Sigma^* | (R_1 \circ R_2)(u) \subseteq X\} \\ &= \{u \in \Sigma^* | R_1(R_2(u)) \subseteq X\} \\ &= \{u \in \Sigma^* | \forall v \in R_2(u) (R_1(v) \subseteq X)\} \\ &= \{u \in \Sigma^* | R_2(u) \subseteq \{v \in \Sigma^* | R_1(v) \subseteq X\}\} \\ &= \{u \in \Sigma^* | R_2(u) \subseteq [R_1] X\} \\ &= [R_2] [R_1] X \end{aligned}$$

But $[R_1]\,X$ is regular language since R_1 is regularity preserving. Therefore, it follows from the fact that R_2 is regularity preserving that $[R_1 \circ R_2]\,X$ is a regular language.

QED

The negative results concerning closure properties now follow:

Theorem 3.4.7 *The class of regularity preserving relations is* **not closed** *under* **intersection, complementation** *or* **inverse**.

Proof. Let $R_1 = \left\{\left(y^k x^n, x^n\right) | n, k \geq 0\right\}$ and let $R_2 = \left\{\left(y^n x^k, x^n\right) | n, k \geq 0\right\}$. Then both R_1 and R_2 are rational relations and therefore regularity preserving.

1. Intersection: The relation $R_1 \cap R_2 = \{(y^n x^n, x^n) | n \geq 0\}$ is not rational and thus it does not necessarily follow that it is regularity preserving. In fact it is not, since
$$\langle R_1 \cap R_2 \rangle \{x\}^* = \{y^n x^n | n \geq 0\}$$
which is not a regular language.

2. Complementation: Although rational relations are generally not closed under complementation, the complements of R_1 and R_2 are both rational relations:
$\overline{R_1} = \Sigma^* \times (\Sigma^* \setminus \{x\}^*) \cup (\epsilon, x)^* (x, \epsilon) ((x, \epsilon) + (y, \epsilon))^* (y, \epsilon) ((x, \epsilon) + (y, \epsilon))^* \cup (\epsilon, x)^* ((x, \epsilon) + (y, \epsilon))^* (y, \epsilon) \cup (y, \epsilon)^* (x, x)^* (\epsilon, x)^+ (y, \epsilon) ((x, \epsilon) + (y, \epsilon))^* \cup (\epsilon, x)^+ \cup (y, \epsilon)^* (x, x)^* ((x, \epsilon) + (y, \epsilon))^+ \cup (y, \epsilon)^* (x, x)^* (\epsilon, x)^+$
$\overline{R_2} = \Sigma^* \times (\Sigma^* \setminus \{x\}^*) \cup (x, \epsilon) ((x, \epsilon) + (y, \epsilon))^* (\epsilon, x)^* \cup (y, x)^* (y, \epsilon)^+ ((x, \epsilon) + (y, \epsilon))^* \cup (y, x)^* (\epsilon, x)^+ (x, \epsilon)^+ ((x, \epsilon) + (y, \epsilon))^* \cup (\epsilon, x)^+ \cup (y, x)^* (x, \epsilon)^* (y, \epsilon) ((x, \epsilon) + (y, \epsilon))^*$
Since rational relations are closed under union, the relation $\overline{R_1} \cup \overline{R_2}$ is again a rational relation and therefore regularity preserving. But $\overline{R_1} \cup \overline{R_2} = \overline{R_1 \cap R_2}$ which means that $\overline{R_1 \cap R_2}$ is regular and by 1 above, the complement of $\overline{R_1} \cup \overline{R_2}$, i.e. $R_1 \cap R_2$ is not regularity preserving.

3. Inverse: Consider the inverse of the regularity preserving relation given in example 3.4.4: $\overline{R} = \{(y^n x^n, x^n) | n \geq 0\}$. This is the same relation as in 1 which is not regularity preserving.

QED

3.4.1 An algebraic view

These two classes of rational relations and regularity preserving relations enjoy a number of closure properties under some opererations and naturally give rise to algebraic structures. Two new types of relational algebra can now be defined.

Definition 3.4.8 *A* **regularity preserving algebra** *is an algebra* $\mathcal{A} = (\mathsf{R}, +, ;, I)$ *satisfying the axioms: for* $R_1, R_2, R_2 \in \mathsf{R}$*:*

1. $R_1; (R_2; R_3) = (R_1; R_2)\,; R_3$
2. $R_1; I = R_1 = I; R_1$
3. $(R_1 + R_2)\,; R_3 = R_1; R_3 + R_2; R_3$

Proposition 3.4.9 *The set regularity preserving relations and with the operations '+' and ';' interpreted as union and composition respectively and* I *is the identity relation, form a regularity preserving algebra.*

Definition 3.4.10 *A* **rational algebra** *is an algebra* $\mathcal{A} = \left(\mathsf{R}, +, ;, ^{\dagger}, I\right)$ *such that* $(\mathsf{R}, +, ;, I)$ *is a regularity preserving algebra and the additional axioms are satisfied: for* $R_1, R_2, R_2 \in \mathsf{R}$

1. $\left(R_1^{\dagger}\right)^{\dagger} = R_1$
2. $(R_1 + R_2)^{\dagger} = R_1^{\dagger} + R_2^{\dagger}$
3. $(R_1; R_2)^{\dagger} = R_2^{\dagger}; R_1^{\dagger}$

Proposition 3.4.11 *The set of rational relations forms a rational algebra where '+', ';' and 'I' interpreted as in Proposition 3.4.9 and* † *interpreted as the inverse.*

These definitions are needed since neither the regularity preserving relations nor the rational relations form a Boolean algebra (it is not closed under intersection and complementation) and consequently neither of these classes form *relation algebras* as defined in [22].

Having considered the diamond modal operations and their duals, it becomes apparent that an algebra describing a class of relations' interaction with sets will be useful. Besides the modal operators, set operators on sets and relations include:

1. **Domain**: $D = \{x | \exists y\,(xRy)\}$
2. **Range**: $W = \{y | \exists x\,(xRy)\}$
3. **Image**: $R\,(X) = \{y | \exists x \in X\,(xRy)\}$

4. **Pierce Product**: $R : (X) = \{x | \exists y \in X\, (xRy)\}$

In fact, the Pierce product is equivalent to the diamond operation. Moreover, the other operations can be defined in terms of the Pierce-product [22].

In [21] algebras of relations interacting with sets via the Pierce product, called **boolean modules**, are introduced. The regularity preserving relations as well as the rational relations can however not be used to form boolean models since neither of these classes of relations form *relation algebras*.

Instead, two new two-sorted algebras are defined below.

Definition 3.4.12 *A* **regularity preserving- (respectively, rational-) Boolean algebra** *is a two-sorted algebra $\mathcal{A} = (\mathcal{R}, \mathcal{S}, :)$ where $\mathcal{R}$ is a regularity preserving (rational) algebra, $\mathcal{S}$ is a boolean algebra and ':' is a mapping $\mathcal{R} \times \mathcal{S} \to \mathcal{S}$ denoted by $R : X$ such that for any $R_1, R_2 \in \mathsf{R}$ and any $X_1, X_2 \in \mathsf{S}$ the following holds:*

1. $R_1 : (X_1 + X_2) = R_1 : X_1 + R_1 : X_2$
2. $(R_1 + R_2) : X_1 = R_1 : X_1 + R_2 : X_1$
3. $R_1 : (R_2 : X_1) = (R_1; R_2)\, X_1$

Proposition 3.4.13 *The class of regularity preserving (respectively, rational) relations over a finite alphabet Σ, together with the class of regular sets over Σ (which is a Boolean algebra), form a regularity preserving- (rational-) Boolean algebra.*

This now enables one to reason about these classes of relations algebraically.

3.5 Rationally presentable structures

When talking about structures, it is important to distinguish between abstract structures, and structures with a certain presentation. The same abstract structure may exhibit different properties in different presentations.

Example 3.5.1 Recall the counterexample given in the previous section to illustrate that regularity preserving relations are not closed under intersection i.e. the relation $R = \{(x^n, y^n x^n)\, | n \geq 0\}$. The fact that this relation is not included in the class of rational relations, greatly depends on the language used and the words assigned to each state in the structure which is equipped with this relation. Let $\mathcal{S}$ be the abstract structure under consideration, then figure 3.15 gives this graph with the presentation discussed above.

If the states are given labels from a different alphabet, or even just different labels from the same alphabet, the resulting relations may be rational. For example, the following presentation can be given to $\mathcal{S}$: let $R = \{(x^n, y^n)\, | n \geq 0\}$. Figure 3.16 describes this graphically.

Figure 3.15: A presentation of $\mathcal{S}$ which is not rational.

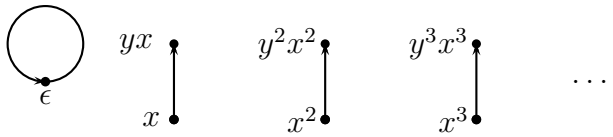
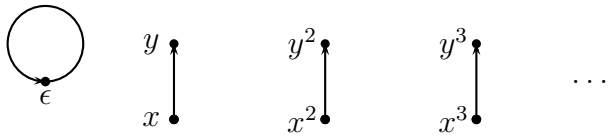
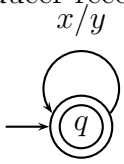


Figure 3.16: A rational presentation of $\mathcal{S}$.



Observe that this graph has the same structure as the previous one - the states have simply been relabeled. To see that this graph is rational observe that it is recognised by the transducer in figure 3.17.

Figure 3.17: The transducer reconising $R = \{(x^n, y^n) \,|n \geq 0\}$.



$\triangleleft$

In the example above, it is apparent that an abstract structure may have a rational presentation - though not all of its presentations may be rational. The term presentation here refers to the labeling of elements of the abstract structure with words over some alphabet. This calls for the introduction of a new notion that will handle this property.

Definition 3.5.2 *An abstract structure $\mathcal{S} = \langle D, R_1, \ldots, R_n \rangle$ is said to be* ***rationally presentable*** *if it is isomorphic to some rational Kripke frame (rational graph).*

Thus, the abstract structure in example 3.5.1 is rationally presentable, since it is isomorphic to the rational Kripke frame given in figure 3.16

Distinguishing between these concepts also effects the way relations and their properties are considered. Rational relations have been defined in Chapter 2, but that was within the presentation of the finitely generated free monoid obtained from some finite alphabet. A stronger property can now be identified for relations of abstract structures.

Definition 3.5.3 *A relation in an abstract structure, is* ***intrinsically rational*** *if, for any regular presentation of the structure, the relation is rational in that presentation.*

Example 3.5.4 Every finite relation is intrinsically rational. ◁

In [69] Morvan refers to the concept of a graph being rationally presentable as being **structurally rational**. He provides an external characterization of rational graphs which produces graphs without assigning names to the vertices. This is done up to isomorphism.

The following will be needed for Morvan's alternate characterisation.

Definition 3.5.5 ([69]) *A context-free language $X \subseteq \Sigma^*$ (see Chapter 2 page 7) is a* **linear language** *if it is generated by a grammar which has at most one non-terminal symbol on the right hand side of each of its rules.*

Definition 3.5.6 *Given a graph $\mathcal{G} = (S, R)$ and a language X, the* **restriction** *of $\mathcal{G}$ to X is the graph $\mathcal{G}_{|X} = \left(S_{|X}, R_{|X}\right)$ where $S_{|X} = S \cap X$ and $R_{|X} = R \cap (X \times X)$.*

If the language X is a regular language, the restriction is called **rational**.

Definition 3.5.7 *A* ***substitution*** *over a free monoid Σ^* is a morphism $\alpha : \Gamma^* \to \mathcal{P}(\Sigma^*)$. Moreover, it maps a language $L \subset \Sigma^*$ to each letter in Γ.*

Now consider the complete binary tree, Λ, given in example 3.1.4. It is labeled by $\Sigma_1 = \{a, b\}$. In addition use the new alphabet $\bar{\Sigma}_2 = \left\{\bar{a}, \bar{b}\right\}$ to label the inverse arcs, i.e. x is related to y by $R_{\bar{a}}$ if and only y is related by x by R_a.

Given a substitution $\alpha : \Gamma^* \to \mathcal{P}(\Sigma_1 \cup \Sigma_2)^*$ define the graph $\alpha^{-1}(\Lambda)$ by:

$$\alpha^{-1}(\Lambda) = \{uR_d w | d \in \Gamma, \text{ there is a path in } \Lambda \text{ labeled by } \alpha(d) \text{ from } u \text{ to } w\}$$

Now let $\mathcal{L}$ be a language and define

$$\mathcal{L}_\Lambda = \{v | \text{ there is a path in } \Lambda \text{ labeled by a word in } \mathcal{L} \text{ from the root} r \text{ to } v\}.$$

Then the graph $\alpha^{-1}(\Lambda)_{|\mathcal{L}_\Lambda}$ is the image of an inverse substitution of the complete binary tree followed by a restriction.

Finally, if α is a substitution such that $\alpha(d)$ is a linear language for every $d \in \Gamma$ then α is called a **linear substitution**. Furthermore, if the $\alpha(d)$ is a linear language such that the projections over barred and non-barred letters are rational for every $d \in \Gamma$, then the languages are called **rational-linear** and the substitution is also rational-linear.

Theorem 3.5.8 ([27]) *A graph is rationally presentable if and only if it is isomorphic to a graph obtained from the complete binary tree Λ by an inverse rational-linear substitution, followed by a rational restriction.*

3.6 Model constructions

Model checking of reachability in rational graphs is generally undecidable - as discussed earlier in this chapter. This means that model checking of the formula $\langle R\rangle^* p$ in a given rational Kripke model as an input is undecidable. Since reachability is the single most important property in infinite-state verification, this fact limits essentially the usability of rational Kripke models for representation and symbolic verification of reachability in infinite transition systems. One way to deal with this problem, is to consider model constructions that do not affect truth of modal formulas expressing reachability. This may enable one to effectively reduce the problem of model checking reachability to model checking reachability in better behaved models where reachability can be computed effectively. In [12] disjoint unions, generated submodels, bounded morphisms and ultrafilter extensions are described as "the big four" operations on Kripke models that do not effect modal satisfaction. A natural question to ask is whether the class of rational Kripke models is closed under these operations.

I will answer questions on the first two of these operations as well as the unfolding of models.

3.6.1 Disjoint unions

Definition 3.6.1 *Two models are disjoint if their domains are disjoint. Let $\mathcal{M}_i = (W_i, R_i, V_i)$ be disjoint models for $i \in I$. Their **disjoint union** is the model $\bigcup_i \mathcal{M}_i = (W, R, V)$ where W is the union of the sets W_i, R is the union of the relations R_i and for each $p \in \Phi$, $V(p) = \bigcup_{i\in I} V_i(p)$. In the case of frames, simply ignore the clause concerning valuations.*

Question 3.6.2 *Is the finite disjoint union of rational Kripke models again a rational Kripke model?*

If the set I of indices is finite, then the answer to this question is 'yes'. Since the domains are disjoint, it is safe to assume without the loss of generality that the domains contain words over disjoint alphabets, i.e. $W_i \subseteq \Sigma_i^*$ for $i \in I$. Suppose $W_i \in \mathsf{REG}(\Sigma_i^*)$ and $W_j \in \mathsf{REG}(\Sigma_j^*)$ and $\Sigma = \Sigma_1 \cup \Sigma_2$ for $i, j \in I$ and $i \neq j$. Then $W_i, W_j \subseteq \Sigma^*$ and moreover they are regular since any regular expression of an alphabet is also a regular expression of an extended alphabet.

An alphabet can now be constructed such that for each $i \in I$ the set W_i is a regular set over this alphabet. In this same way the rational relations will also be rational

relations of the extended alphabet. The claim follows from the closure of regular sets and rational relations under finite union.

3.6.2 Generated substructures

Definition 3.6.3 *Let $\mathcal{F} = (W, R)$ and $\mathcal{F}' = (W', R')$ be two Kripke frames, then $\mathcal{F}'$ is a **subframe** of $\mathcal{F}$ if $W' \subseteq W$ and R' is the restriction of R to W' i.e. $R' = R \cap (W' \times W')$. $\mathcal{F}'$ is a **generated subframe** of $\mathcal{F}$ if $\mathcal{F}'$ is a subframe of $\mathcal{F}$ and for all points $w \in W$ the following closure condition holds:*

$$\textit{If } w \in W' \textit{ and } wRv \textit{ then } v \in W'.$$

*For a model $\mathcal{M}' = (\mathcal{F}', V')$ to be a **submodel** of a model $\mathcal{M} = (\mathcal{F}, V)$ the frame $\mathcal{F}$ must be a subframe of $\mathcal{F}$ and V' is the restriction of V to W' i.e. for each $p \in \Phi$, $V'(p) = V(p) \cap W'$. Thus, $\mathcal{M}'$ is a generated submodel of $\mathcal{M}$ if it is a submodel and it satisfies the closure condition.*

Question 3.6.4 *Given a rational set X, is the submodel (subframe) of a rational Kripke model (frame) generated by X again a rational model (frame)?*

Proposition 3.6.5 *Given a rational set X, then it is not necessarily the case that the subframe (submodel) of a rational Kripke frame (model) generated by X is again a rational Kripke frame (model).*

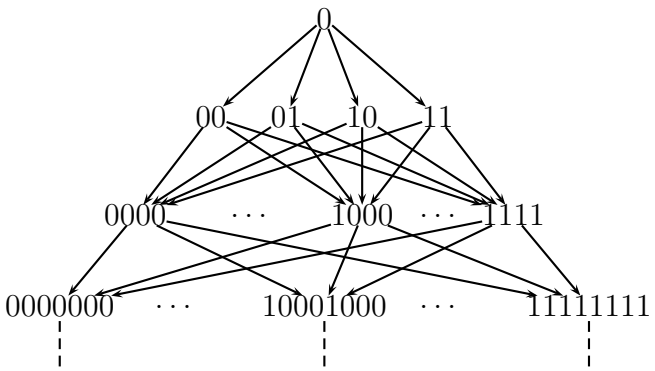
Proof. Consider the rational Kripke frame $\mathcal{F}$, given in Example 3.1.17 on page 29. Let $X = \{0\}$. Then the subframe of $\mathcal{F}$ generated by X is given in figure 3.18. The set of states of the generated subframe is the set $Y = \{u \in \{0,1\}^* \mid l(u) = 2^i, i > 0\} \cup \{0\}$, containing words with length equal to a power of 2. Suppose Y is regular, then by the pumping lemma, it is possible to express a word u of sufficient length in terms of some factors u_1, v and u_2 such that every word in $u_1 v^* u_2$ belongs to Y. Since v must be non-empty, it is easy to see that any such expression will produce words which are not of length equal to some power of 2 and therefore not in Y. Therefore Y is not regular. Since the domain of a rational relation is regular (Theorem 2.5.7), the relation of the generated subgraph cannot be rational.

QED

3.6.3 Unfoldings

Consider the labeled graph $\mathcal{G} = \left(S, \{E_a\}_{a \in \Gamma}\right)$ where Γ is some finite alphabet of labels. If the set S is finite then the graph is finite, and since every finite set is a regular set, the graph $\mathcal{G}$ is a rational graph and therefore a rational Kripke frame. Now, a labeled path in $\mathcal{G}$ is a sequence $\mathsf{p} = (v_0, a_1, v_1, \ldots, a_k, v_k)$ where $a_i \in \Gamma$, $v_i \in S$ and $v_{i-1} E_{a_i} v_i$

Figure 3.18: The subframe generated by the regular set $X = \{0\}$.



for $i = 1, 2, \ldots, k$. Let $f(\mathsf{p})$ denote the final vertex of the path p. Lastly, denote the set of all paths in the graph $\mathcal{G}$ starting at a vertex v_0 by $\mathsf{P}(v_0)$.

Definition 3.6.6 *The tree unfolding of a graph $\mathcal{G} = \left(S, \{E_a\}_{a\in\Gamma}\right)$ is the labeled graph $\mathcal{T}(\mathcal{G}, v_0) = \left(\mathsf{P}(v_0), \{E_a^p\}_{a\in\Gamma}\right)$ where the relation E_a^p is defined by:*

$$E_a^p := \{(\mathsf{p}, \mathsf{p} \bullet (a, u)) \,|\, \mathsf{p} \in \mathsf{P}(v_0), f(\mathsf{p})\, E_a u\}$$

where the operation denoted by $\bullet$ is path concatenation.

Question 3.6.7 *Is the unfolding of a finite Kripke frame always a rational Kripke frame?*

The answer to this question is 'yes' as explained below.

Definition 3.6.8 *A tree is a **regular tree** if and only if it has only finitely many non-isomorphic subtrees.*

Proposition 3.6.9 *The unfolding of a finite Kripke frame is a rational Kripke frame.*

Proof. While studying Caucal's hierarchy, Thomas observes in [87] that the set of tree unfoldings of finite graphs is the set of regular trees.

Following Caucal's hierarchy, the class of regular trees are included in the class of prefix-recognisable graphs ([87]). However, in [69] Morvan observes that prefix recognisable graphs are included in the class of rational graphs proving the claim. QED

In the case of rational Kripke models it is necessary to first describe how the tree unfolding effects the valuation.

Let $\mathcal{M} = (\mathcal{G}, V)$ be a rational Kripke model based on the rational Kripke frame $\mathcal{G}$, then the tree unfolding of $\mathcal{M}$ is the Kripke model given by

$$\mathcal{T}(\mathcal{M}, v_0) = (\mathcal{T}(\mathcal{G}, v_0), V_1)$$

where the valuation V_1 is given by

$$V_1(p) = \{\mathsf{p} | f(\mathsf{p}) \in V(p)\}$$

for $p \in \Phi$.

Question 3.6.10 *Is the unfolding of a finite Kripke model always a rational Kripke model?*

Again the answer is 'yes'. It has already been shown that tree unfolding preserves rational Kripke frames. Therefore, all that remains to establish is whether or not tree unfoldings preserve the rationality of the valuations.

The valuation V_1 of a propositional variable contains all the finite paths in a finite graph that end at a state in the regular set $V(p)$. In fact, the set $V(p)$ is finite since the graph is finite. For each state in $V(p)$ the set of paths ending in it is regular since a regular expression can be given for it: It is the union of a finite number of regular sets of the form: $v_0((a_1v_1)^* + (a_1v_1)) \ldots ((a_kv_k)^* + (a_kv_k))$ where $v_k \in V(p)$ and $(a_iv_i)^*$ is in the expression if the edge forms a loop. There are finitely many since there are only finitely many edges in $\mathcal{G}$ to follow. Thus, $V_1(p)$ is a finite union of regular sets and therefore regular itself.

3.7 Bisimulation

Another very important type of relation between models is bisimulation. In fact, the bisimulation preserves the satisfaction of the entire modal μ-calculus - making it a particularly important type of relation to study.

The work in the following two subsections also appears in [6].

3.7.1 Bisimulation between two rational Kripke structures.

Consider the following two algorithmic problems:

Question 3.7.1 *Are two given pointed rational Kripke frames (respectively, models) bisimilar?*

Question 3.7.2 *Are two given rational Kripke frames (respectively, models) globally bisimilar?*

Definition 3.7.3 *A counter machine* C *is a program containing non-negative counters* $c_1, c_2, \ldots, c_m$ *and commands* $S_1, S_2, \ldots, S_n$*, where* S_n *is the halting command and for* $1 \leq i < n$ *the command* S_i *is of the form:*

$$S_i : \ c_j := c_j + 1; \quad go \ to \ \ k$$

or

$$S_i : \ if \, c_j = 0 \ \ go \ to \ \ k_1 \, else \, c_j := c_j - 1 \, and \, go \ to \ \ k_2$$

for $1 \leq k, k_1, k_2 \leq n$*,* $1 \leq j \leq n$ *and '*$:=$*' denoting assignment. A counter machine starts off performing command* S_1*.*

Proposition 3.7.4 ([6]) *It is undecidable whether two given pointed rational Kripke models are bisimilar.*

Proof. Firstly recall that the configuration graphs of Petri nets are rational Kripke frames (Proposition 3.1.19). In [49], Jančar proved that global bisimilarity is undecidable for labeled Petri nets by constructing, for a fixed (universal) 2-counter machine and a pair of input values, a pair of labeled Petri nets such that their configuration graphs are bisimilar if and only if the counter machine does not halt for the given input. The claim then follows by constructing two rational Kripke models based upon the configuration graphs of the Petri nets from Jančar's proof.

I will now present the details of Jančar's construction.

Let C be a fixed (universal) counter machine with two counters c_1, c_2 and commands $S_1, S_2, \ldots, S_n$, for which the halting problem is undecidable. A net that simulates C can now be constructed.

Constuction 3.7.5 *Let* $N = (P, T, F, L)$ *be the labeled net obtained as follows:*

Let $P = \{c_1, c_2, s_1, \ldots, s_n, p, p'\}$*. Assume that all of the places are empty, i.e. they don't contain any tokens. Adding an arc* (q, t) *to the net means to increment* $F(q, t)$ *with one. Begin with* $T = \emptyset$ *and for each transition* t *added, let* $F(q, t) = 0$ *and* $F(t, q) = 0$ *for all* $q \in P$*. Transitions and arcs will now be added to the net using the following rules: For* $1 \leq i < n$*,* $1 \leq k, k_1, k_2 \leq n$ *and* $1 \leq j \leq n$

1. *If command* S_i *is of the form*

$$S_i : \ c_j := c_j + 1; \quad go \ to \ \ k$$

then add the transition t_i *to the net, as well as the arcs* $(s_i, t_i), (t_i, c_j)$ *and* (t_i, s_k)*. Let the label be* $L(t_i) = a_i$

2. *If command* S_i *is of the form*

$$S_i : \ if \, c_j = 0 \, goto \, k_1 \, else \, c_j := c_j - 1 \ \ and \, go \, to \, k_2$$

(a) add transition t_i^{NZ} *and arcs* $\left(s_i, t_i^{NZ}\right), \left(c_j, t_i^{NZ}\right)$ *and* $\left(t_i^{NZ}, s_{k_2}\right)$ *to the net. Let* $L\left(t_i^{NZ}\right) = a_i^{NZ}$.

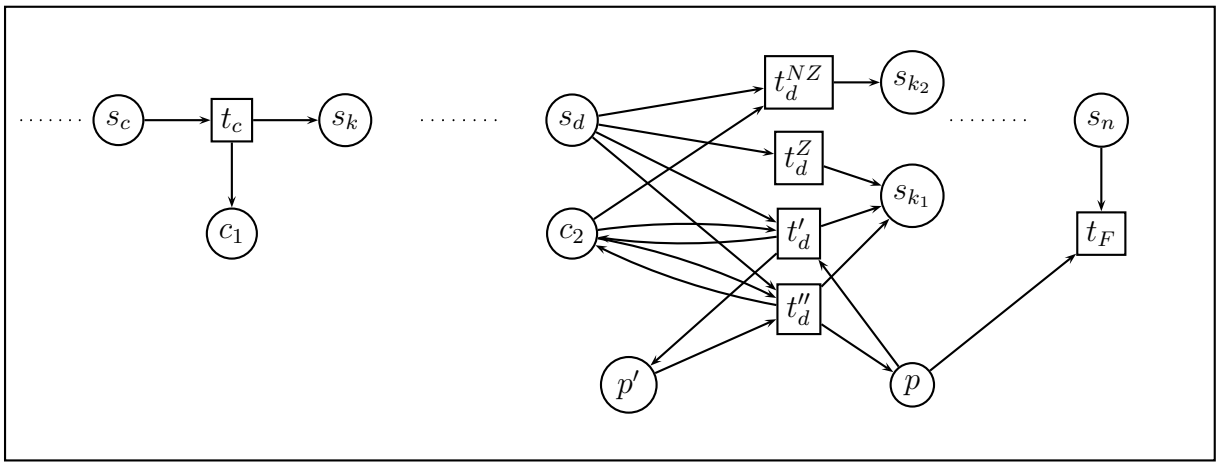
(b) add transitions t_i^Z, t_i' *and* t_i'' *to the net. Also add the following arcs:*

i. $\left(s_i, t_i^Z\right)$ *and* $\left(t_i^Z, s_{k_2}\right)$
ii. $\left(p, t_i'\right), \left(t_i', p'\right), \left(p', t_i''\right)$ *and* $\left(t_i'', p\right)$
iii. $\left(s_i, t_i'\right), \left(s_i, t_i''\right), \left(t_i', s_{k_1}\right), \left(t_i'', s_{k_1}\right), \left(t_i', c_j\right), \left(t_i'', c_j\right), \left(c_j, t_i'\right)$ *and* $\left(c_j, t_i''\right)$

Let $L\left(t_i^Z\right) = L\left(t_i'\right) = L\left(t_i''\right) = a_i$.

3. Lastly, add the transition t_F *with the label* a_F *and the arcs* (s_n, t_F) *and* (p, t_F).

Figure 3.19: The net N constructed to simulate C. Suppose S_c is a command of the first type using c_1 and S_d of the second type using c_2 where $1 \leq c, d < n$.



Jančar showes that bisimilarity is undecidable for the Petri nets $\mathcal{N}_1 = (N, M_1)$ and $\mathcal{N}_2 = (N, M_2)$, where N is the net constructed above and M_1, M_2 are the markings

$$M_1 = \left(x_1, x_2, \underbrace{1, 0, \ldots, 0}_{n}, 1, 0\right)$$

$$M_2 = \left(x_1, x_2, \underbrace{1, 0, \ldots, 0}_{n}, 0, 1\right)$$

where the places are arranged in the order $\{c_1, c_2, s_1, \ldots, s_n, p, p'\}$. The values x_1 and x_2 are the input values for C.

Observe that the only difference between these two Petri nets is their initial markings. Hence, their configuration graphs will be exactly the same graph. Let C be that graph. Then it is undecidable whether the pointed rational Kripke frames (C, M_1) and (C, M_2) are bisimilar or not. QED

In [50] Jančar and Srba prove the undecidability of bisimilarity for prefix rewrite systems. Since prefix recognisable graphs are rational [29] this serves as another proof of the undecidability of bisimilarity for rational Kripke frames. In [50] the authors reduced a version of the Post's correspondence problem to the bisimilarity problem for prefix rewrite systems.

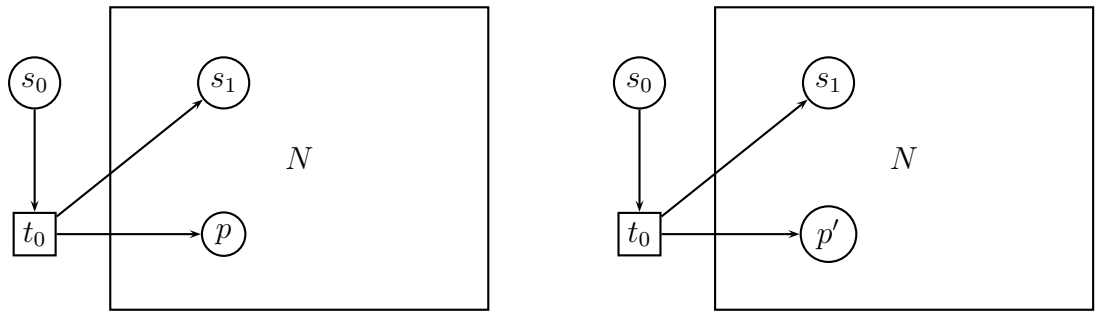
This result now carries over to global bisimilarity:

Corollary 3.7.6 ([6]) *It is undecidable whether two given rational Kripke frames are globally bisimilar.*

Proof. The basic idea is that the bisimilarity of the pointed Kripke frames produced from the configurations graphs of the Petri nets in Jančar's proof can be reduced to the global bisimilarity of two Kripke frames. This can be done by adding a unique appendix to each of the distinguished points of the two pointed Kripke frames, such that every global bisimulation between the resulting frames has to map these appendices to each other. A bisimulation on the respective pointed frames is then produced, and vice versa - every such bisimulation extends to a global bisimulation between the Kripke frames with the appendices.

Constuction 3.7.7 *Let N be the net constructed to simulate* C. *Construct the net N_1 by adding the place s_0, the transition t_0 and the arcs $(s_0, t_0), (t_0, s_1)$ and (t_0, p) to N. N_2 is constructed by adding s_0, t_0 and the arcs $(s_0, t_0), (t_0, s_1)$ and (t_0, p') to N. In both nets let $L(t_0) = a_0$.*

Figure 3.20: The newly constructed nets N_1 and N_2.



Let M be the set of all finite words over the alphabet $\{c_1, c_2, s_0, s_1, \ldots, s_n, p, p'\}$. Furthermore, for both N_1 and N_2 let R_t be the rational relation corresponding to the transition $t \neq t_0$. Since the behaviour of the transitions not equal to t_0 are the same in both nets, the corresponding rational relations coincide. The different R_t's are defined as follows where $1 \leq c, d < n$

1. For t_c where the command S_c is of the first type and for instance using the counter c_1

$$R_{t_i} = \left\{ c_1^{x_1} c_2^{x_2} s_0^0 s_1^0 \ldots s_c^1 \ldots s_n^0 p^r p'^s \xrightarrow{t_c} c_1^{x_1+1} c_2^{x_2} s_0^0 s_1^0 \ldots s_c^0 \ldots s_k^1 \ldots s_n^0 p^r p'^s \right\}$$

   with $r + s = 1$.

2. For S_d a command of type two and for instance using counter c_2 let $r + s = 1$ again.

   (a) The transition t_d^{NZ} induces the relation

$$R_{t_d^{NZ}} = \left\{ c_1^{x_1} c_2^{x_2} s_0^0 s_1^0 \ldots s_d^1 \ldots s_n^0 p^r p'^s \xrightarrow{t_d^{NZ}} c_1^{x_1} c_2^{x_2-1} s_0^0 s_1^0 \ldots s_d^0 \ldots s_{k_2}^1 \ldots s_n^0 p^r p'^s \right\}$$

   (b) t_d^Z induces

$$R_{t_d^Z} = \left\{ c_1^{x_1} c_2^{x_2} s_0^0 s_1^0 \ldots s_d^1 \ldots s_n^0 p^r p'^s \xrightarrow{t_d^Z} c_1^{x_1} c_2^{x_2} s_0^0 s_1^0 \ldots s_d^0 \ldots s_{k_1}^1 \ldots s_n^0 p^r p'^s \right\}$$

   (c) For $x_2 > 0$ the relation induced by t'_d is

$$R_{t'_d} = \left\{ c_1^{x_1} c_2^{x_2} s_0^0 s_1^0 \ldots s_d^1 \ldots s_n^0 p^1 p'^0 \xrightarrow{t'_d} c_1^{x_1} c_2^{x_2} s_0^0 s_1^0 \ldots s_d^0 \ldots s_{k_1}^1 \ldots s_n^0 p^0 p'^1 \right\}$$

   (d) The relation induced by t''_d where $x_2 > 0$ is

$$R_{t''_d} = \left\{ c_1^{x_1} c_2^{x_2} s_0^0 s_1^0 \ldots s_d^1 \ldots s_n^0 p^0 p'^1 \xrightarrow{t''_d} c_1^{x_1} c_2^{x_2} s_0^0 s_1^0 \ldots s_d^0 \ldots s_{k_1}^1 \ldots s_n^0 p^1 p'^0 \right\}$$

3. Lastly the relation induced by the transition t_F is given by

$$R_{t_F} = \left\{ c_1^{x_1} c_2^{x_2} s_0^0 s_1^0 \ldots s_{n-1}^0 s_n^1 p^1 p'^0 \xrightarrow{t_F} c_1^{x_1} c_2^{x_2} s_0^0 s_1^0 \ldots s_n^0 p^0 p'^0 \right\}$$

The valid markings are considered to be those for which at most one of the s_i's, for $1 \leq i \leq n$, contains one token and the sum of the tokens in p and p' is 1. This is because the aim for the net N is to simulate the behaviour of the counter machine which can

only obey one command at a time. Furthermore, only after the final transition has been fired, do none of the places $s_1, \ldots, s_n, p, p'$ contain any tokens.

Now let x, y be specific input values for the machine C and set

$$R^1_{t_0} = \left\{ c_1^x c_2^y s_0^1 s_1^0 \ldots s_n^0 p^0 p'^0 \xrightarrow{t_0} c_1^x c_2^y s_0^0 s_1^1 \ldots s_n^0 p^1 p'^0 \right\}$$

and

$$R^2_{t_0} = \left\{ c_1^x c_2^y s_0^1 s_1^0 \ldots s_n^0 p^0 p'^0 \xrightarrow{t_0} c_1^x c_2^y s_0^0 s_1^1 \ldots s_n^0 p^0 p'^1 \right\}$$

$R^1_{t_0}$ and $R^2_{t_0}$ are rational relations (they are singletons) and thus so are the relations $R_1 = R^1_{t_0} \cup \bigcup_{t \neq t_0} R_t$ and $R_2 = R^2_{t_0} \cup \bigcup_{t \neq t_0} R_t$ of the nest N_1 and N_2 respectively. Consequently the configuration graphs of the nets N_1 and N_2, say C_1 and C_2, are rational Kripke frames.

Firstly, consider the word $u = c_1^x c_2^y s_0^1 s_1^0 \ldots s_n^0 p^0 p'^0$ in C_1. Clearly, there is no bisimulation relating any word $v \neq u$ in C_2 to u since u has an a_0 successor and v not (the forth condition is broken). Hence the only word in C_2 a bisimulation can possibly relate to u is the word u. A bisimulation relates u in C_1 to u in C_2 if and only if the successor $v_1 = c_1^x c_2^y s_0^0 s_1^1 \ldots s_n^0 p^1 p'^0$ of u in C_1 (i.e. its only successor in C_1) is related to the successor $v_2 = c_1^x c_2^y s_0^0 s_1^1 \ldots s_n^0 p^0 p'^1$ of u in C_2 (again the only one there is). In these successors the value representing the number of tokens in s_0 is zero and there are no transitions in either N_1 or N_2 that increase the number of tokens in s_0. Hence, whether or not v_1 and v_2 are related by a bisimulation only depends on the transitions in N_1 and N_2 that coincide - which also coincides with N's transitions. Consequently there is a bisimulation relating v_1 and v_2 if and only if the words $c_1^x c_2^y s_1^1 \ldots s_n^0 p^1 p'^0$ and $c_1^x c_2^y s_1^1 \ldots s_n^0 p^0 p'^1$ in C are related by a bisimulation, i.e. if and only if the pointed Kripke models are bisimilar. The latter problem is however undecidable.

Next, any word in C_1 that does not represent a valid marking will be related to its counterpart in C_2 by any bisimulation. In neither of these graph do they have any successors so the back and forth conditions are vacuously satisfied.

Now consider a word $w \neq u$ which represents a valid marking. Firstly w has no a_0 successor. Secondly, u is not in the image of any of the relations $R_{t \neq t_0}$ and thus u is not reachable from w. Hence any path leading from the word w in C_1 will also be a path in C_2 and vice versa. Thus, there will be a bisimulation relating w in C_1 and w in C_2.

In conclusion, whether there exists a global bisimulation between C_1 and C_2 only depends on whether (C_1, v_1) and (C_2, v_2) are bisimilar. Therefore global bisimilarity between C_1 and C_2 is undecidable. QED

Corollary 3.7.8 ([6]) *It is undecidable whether two given pointed rational Kripke models are bisimilar, or whether two given rational Kripke models are globally bisimilar.*

3.7.2 The finite bisimilarity problem

There are two questions to be answered with respect to deciding whether or not a rational Kripke models is finite up to bisimulation - in [39] Jančar called them the equivalence and finiteness problems respectively.

Question 3.7.9 ***The finite equivalence problem:*** *Given an arbitrary pointed rational Kripke model $\mathcal{M}$ and a finite pointed rational Kripke model $\mathcal{N}$, are they bisimilar.*

Question 3.7.10 ***The finite bisimilarity problem:*** *Given a pointed rational Kripke model $\mathcal{M}$, is $\mathcal{M}$ finite up to bisimulation? In other words, does there exist a finite pointed rational Kripke model $\mathcal{N}$ bisimilar to $\mathcal{M}$?*

A pointed rational Kripke model will be called generated if every state in the model is reachable form the the distinguished one.

The first question will be answered here for the case where the pointed model is generated.

Suppose the generated rational Kripke model $\mathcal{M}$ has a finite bisimulation quotient. Is there a way to 'effectively produce' classes of $\mathcal{M}$'s domain? By 'effectively produce' I mean to say that the classes must be symbolically presentable.

The answer is 'yes'. The Paige-Tarjan Algorithm is suitable for this.

A **rational partition** of a regular language L is one in which every subset in the partition is a regular sublanguage of L. If $\mathcal{M} = (M, R, V)$ is a rational Kripke model, the function $W : \mathcal{P}(\Phi) \to \mathcal{P}(M)$ is defined such that for every $P \in \mathcal{P}(\Phi)$

$$W(P) = \{x \in M \mid \forall p \in P\,(x \in V(p)) \wedge \forall p \notin P\,(x \notin V(p))\}$$

Lemma 3.7.11 ([6]) *Let $\mathcal{M} = (M, R, V)$ be a rational Kripke model and Φ the set of propositional variables in the language. Then $\{W(P) \,|P \in \mathcal{P}(\Phi)\}$ is a rational partition of the state space M of $\mathcal{M}$, called* ***the partition of*** *M* ***induced by*** *Φ.*

Proof. For every $P \in \mathcal{P}(\Phi)$ we have that $W(P) = \big(\bigcap_{p\in P} V(p)\big)\backslash\big(\bigcup_{p\notin P} V(p)\big)$. Thus $W(P)$ is a boolean combination of regular sets and therefore regular itself. QED

Proposition 3.7.12 ([6]) *It is (semi-)decidable whether or not a given pointed generated rational Kripke model $(\mathcal{M}_r, u)$ and given finite pointed Kripke model $(\mathcal{M}_f, v)$ are bisimilar.*

Proof. To prove this claim a semi-algorithm which halts if $\mathcal{M}_r$ is finite up to bisimulation is given. Minor alterations are made to the Paige-Tarjan algorithm. The resulting semi-algorithm can then be applied effectively to rational Kripke models.

Algorithm:

Let $\mathcal{M}_r = (\Sigma^*, R, V)$ be the rational Kripke model under consideration, with $(\mathcal{M}_r, s)$ the pointed rational Kripke model.

1. Start off with the rational partitioning of the domain induced by Φ, with the only change being that the vertex u is placed in its own block: Let $P' \in \mathcal{P}(\Phi)$ be the set of exactly all those propositional variables satisfied at u.

$$\mathsf{P}_0 = \{V''(P) \,|P \in \mathcal{P}(\Phi), P \neq P'\} \cup \{S, V''(P') \setminus S\}$$

where $S = \{u\}$. Furthermore, let $\mathcal{P}_{split} \subset \mathsf{P}_0$ and initialise it to $\mathsf{P}_{split} = \{S\}$. Let $k_0 = |\mathsf{P}_{split}|$.

2. To obtain the partition P_i choose an arbitrary set $A_i \in \mathsf{P}_{split}$ and perform the splitting procedure of the sets in P_{i-1}:

$$\mathsf{P}_i = \left\{B \cap R_{-1}(A_i), B \setminus R^{-1}(A_i) \,|B \in \mathsf{P}_{i-1}\right\}$$

and update P_{split}'s content such that:

$$\mathsf{P}_{split} = \mathsf{P}_{split} \cup \left\{B \cap R^{-1}(A_i) \,|B \in \mathsf{P}_{i-1}\right\}$$

and then $\mathsf{P}_{split} = \mathsf{P}_{split} \setminus \{\emptyset\}$. Lastly set $k_i = |\mathsf{P}_{split}|$.

The alteration made to the Paige-Tarjan algorithm is that only sets that contain words reachable form the word u, or u itself are used to perform the splitting - since we are only interested in sets that can be reached from u.

Now, if there is a finite t such that for any of the k_t sets in P_{split} the splitting procedure does not produce a new partition, then a fixpoint is reached.

Each partition P_i is a set of regular sets: P_0 contains only regular sets. A combination of boolean operators are applied to the sets in P_{i-1} and the claim then follows since regular sets are closed under the boolean operations. These closure properties are then also the reason why each partition that follows is a set of regular sets.

Let P_t now be the final partition reached and label its elements from D_1 to D_l. Define a finite rational Kripke model as follows: $\mathcal{MF} = (W_1, R_1, V_1)$ with $W_1 = \{1, 2, \ldots, l\}$ and iR_1j holds for $1 \leq i, j \leq l$ if and only if $D_i \cap R^{-1}(D_j) \neq \emptyset$. Then the relation $\mathsf{B} : \Sigma^* \to W_1$ defined by $s\mathsf{B}i$ for $s \in \Sigma^*$ and $0 \leq i \leq l$ if and only if $s \in D_i$, is a bisimulation from Σ^* to W_1. Hence, $(\mathcal{M}_r, u)$ and $(\mathcal{MF}, w)$ are bisimilar, where $u \in D_w$.

Since two pointed Kripke structures are bisimilar if and only if their bisimulation quotients are isomorphic [47], the pointed model $(\mathcal{M}_r, u)$ will be bisimilar to $(\mathcal{M}_f, v)$, if and only if $(\mathcal{MF}, w)$ is bisimilar to $(\mathcal{M}_f, v)$. Since these two pointed models are both finite, bisimilarity between them is decidable. Thus, if the answer is positive, this procedure will produce it. QED

Definition 3.7.13 The pointed frames (models) $(\mathcal{S}_1, u_1)$ and $(\mathcal{S}_2, u_2)$ are **n-bisimilar** (or, **n-bisimulation equivalent**) if there is an n-bisimulation B between $\mathcal{S}_1$ and $\mathcal{S}_2$ such that B relates u_1 and u_2.

Intuitively, an **n-bisimulation** is a relation between two frames (models) that preserves atomic information and transition possibilities up n steps.

The concept of n-bisimulation can be rephrased in terms of a *bisimulation game*, so that two pointed Kripke frames (models) are n-bisimilar if and only if Player II has a winning strategy in the n-round bisimulation game. For more details see e.g. [47].

Lemma 3.7.14 ([6]) *It is decidable whether or not two given pointed rational Kripke models are n-bisimilar.*

Proof. Two methods to determine whether two given pointed Kripke models $(\mathcal{M}_1, u_1)$ and $(\mathcal{M}_2, u_2)$ are n-bisimilar are given.

First method:

Firstly determine the partitions $\mathcal{P}_1$ and $\mathcal{P}_2$ of M_1 and M_2 induced by Φ. Note that there must be a one-to-one correspondence between the elements in $\mathcal{P}_1$ and $\mathcal{P}_2$. Enumerate the elements in $\mathcal{P}(\Phi)$ and let $\mathsf{B}_i^0 = \{u_i\}$ for $i = 1, 2$. We will construct sequences of regular languages B_i^j for $j = 1, 2, \ldots, n$ (if possible) as follows:

1. Initialize $\mathsf{B}_1^j = \mathsf{B}_2^j = \emptyset$.
2. Enumerate the sets in B_1^{j-1} and B_2^{j-1}, e.g., $\mathsf{B}_1^{j-1} = \{A_1, A_2, \ldots, A_m\}$ and $\mathsf{B}_2^{j-1} = \{C_1, C_2, \ldots, C_m\}$. The lists B_1^{j-1} and B_2^{j-1} must be of equal length and contain matching elements in matching positions. Now, for $k = 1, 2, \ldots, m$:
   (a) Compute the regular language $R(A_k)$ and then the regular languages $R(A_k) \cap W(P_l)$ for every $P_l \in \mathcal{P}(\Phi)$.
   (b) Compute the regular language $R(C_k)$ and then the regular languages $R(C_k) \cap W(P_l)$ for every $P_l \in \mathcal{P}(\Phi)$.
   (c) If $R(A_k) \cap W(P_l) = \emptyset \iff R(C_k) \cap W(P_l) = \emptyset$, iterate through the elements in $\mathcal{P}(\Phi)$, adding the language $R(A_k) \cap W(P_l)$ at the end of B_i^j for each iteration.

   Otherwise B_i^j cannot be obtained. Halt the procedure and return 'no', i.e $(\mathcal{M}_1, u_1)$ and $(\mathcal{M}_2, u_2)$ are not n-bisimilar.

If B_1^n and B_2^n are constructed successfully, return 'yes' i.e. $(\mathcal{M}_1, u_1)$ and $(\mathcal{M}_2, u_2)$ are n-bisimilar.

Second method

Alternatively, *characteristic formulas* (see [47]) can be used to determine whether two models are n-bisimilar. Considering $(\mathcal{M}_1, u_1)$ as the reference structure, we can characterize n-bisimilarity to $(\mathcal{M}_1, u_1)$ with a modal formula $\chi^n_{[\mathcal{M}_1, u_1]}$, such that $(\mathcal{M}_1, u_1)$ and $(\mathcal{M}_2, u_2)$ are n-bisimilar if and only if $\mathcal{M}_2, u_2 \models \chi^n_{[\mathcal{M}_1, u_1]}$. We define $\chi^n_{[\mathcal{M}_1, u_1]}$ inductively as follows:

1. $\chi^0_{[\mathcal{M}_1,u_1]}$ is the conjunction of all $p \in \Phi$ that are true at u_1 and all $\neg p$ that are false at u_1 in $\mathcal{M}_1$.

2. $\chi^{n+1}_{[\mathcal{M}_1,u_1]} := \chi^0_{[\mathcal{M}_1,u_1]} \wedge \bigwedge_{u_1 R u_2} \langle R \rangle \chi^n_{[\mathcal{M}_1,u_1]} \wedge [R] \bigvee_{u_1 R u_2} \chi^n_{[\mathcal{M}_1,u_2]}$.

Now, do local model checking of $\chi^n_{[\mathcal{M}_1,u_1]}$ in $\mathcal{M}_2$ at u_2. If $\mathcal{M}_2, u_2 \models \chi^n_{[\mathcal{M}_1,u_1]}$ return 'yes', else return 'no'.

The second approach has the advantage that, once $\chi^{n+1}_{[\mathcal{M}_1,u_1]}$ is determined, it can be used to determine whether any other given model is n-bisimilar to $(\mathcal{M}_1, u_1)$. QED

Proposition 3.7.15 ([6]) *It is (semi-)decidable whether or not a pointed rational Kripke model $(\mathcal{M}, u)$ and a pointed finite Kripke model $(\mathcal{N}, v)$ are not bisimilar.*

Proof. A semi-algorithm subsequently testing n-bisimulations of $(\mathcal{M}_r, u)$ and $(\mathcal{M}_f, v)$ for $n = 1, 2, \ldots$, using either of the methods in lemma 3.7.14 is run. If the two models are not bisimilar, then they are not n-bisimilar for a large enough n, so this algorithm will terminate after finitely many steps. QED

The two semi-decidability results now have the following interesting consequences:

Corollary 3.7.16 ([6]) *It is decidable whether or not a given pointed generated rational Kripke model $(\mathcal{M}_r, u)$ and given finite pointed Kripke model $(\mathcal{M}_f, v)$ are bisimilar.*

Chapter 4

Subclasses of Rational Kripke Structures

In this chapter three subclasses of rational Kripke structures are investigated. Morvan [71] gives two ways to isolate subclasses of rational graphs. The first way is a structural restriction on the graph itself, while the second involves a limitation on the transducer. This chapter deals with one subclass of the first type and two subclasses of the second.

4.1 Automatic Kripke structures

Definition 4.1.1 ([66]) *A transducer $\mathcal{T} = \langle Q, \Sigma, \Gamma, q_i, F, \rho \rangle$ is called **letter-to-letter** or **synchronous** if every $(p, u, v, q) \in \rho$ satisfies $u, v \in \Sigma$.*

A relation recognised by a synchronous transducer is **length-preserving** since it only relates words $u, v \in \Sigma^*$ if $|u| = |v|$. This is not a very general class of relations and in order to obtain a broader class, the synchronisation of the transducer can be relaxed ([37]).

Definition 4.1.2 ([79]) *Let $\nabla \notin \Sigma$, then $\Sigma_\nabla = \Sigma \cup \{\nabla\}$. The **convolution** of a pair (u, v) is the pair $(u\nabla^i, v\nabla^j)$ from $\Sigma_\nabla \times \Sigma_\nabla$ where $i = \max(0, |v| - |u|)$ and $j = \max(0, |u| - |v|)$. Then the **convolution** of a binary relation R over Σ, denoted by $\otimes R$, is the relation containing all the convolutions of the pairs in R, i.e. $\otimes R = \{(u\nabla^i, v\nabla^j) \mid (u, v) \in R\}$*

Definition 4.1.3 ([79]) *A binary relation $R \subseteq \Sigma^* \times \Sigma^*$ over a finite alphabet Σ is* **regular** *(***automatic** *or* **finite automaton recognisable***) if its convolution $\otimes R$ is recognisable by a synchronous transducer.*

Definition 4.1.4 ([79]) *An **automatic Kripke frame** is a rational Kripke frame $\mathcal{F} = (S, R)$ over a finite alphabet Σ such that $S \subseteq \Sigma^*$ is a regular language and $R \subseteq$*

$\Sigma^* \times \Sigma^*$ *is regular.*

Example 4.1.5 The transducer $\mathcal{T}$ in figure 4.1 is a synchronous transducer over $\Sigma_\nabla = \{0, 1, \nabla\}$. Consider the Kripke frame $\mathcal{F} = (S, R)$ where $S = (0+1)^* 0$ and $R = R_1 \cup R_2 \cup R_3$ with:

$$R_1 = (0,1)\,(\epsilon, u)\,(\epsilon, 0)^+$$

$$R_2 = (1,1)\,(u, v)\,(0, w)\,(\epsilon, 0)^+$$

$$R_3 = (1,1)\,(u, u)\,(0, 1)\,(v, w)\,(0, 0)^+$$

for $u, v, w \in \Sigma^*$.

Then $\mathcal{T}$ recognises $\otimes R$ and the Kripke frame $\mathcal{F}$, given in figure 4.2, is automatic. $\mathcal{F}$ is the frame where each even number $n \in \mathbb{N}$ is encoded in the standard way as a binary string without leading zeros and each encoded even number is related to all the encodings of even numbers strictly greater than itself. The automatic Kripke frame $\mathcal{F}$ does not have a bounded degree.

Figure 4.1: The synchronous transducer $\mathcal{T}$ recognising $\otimes R$.

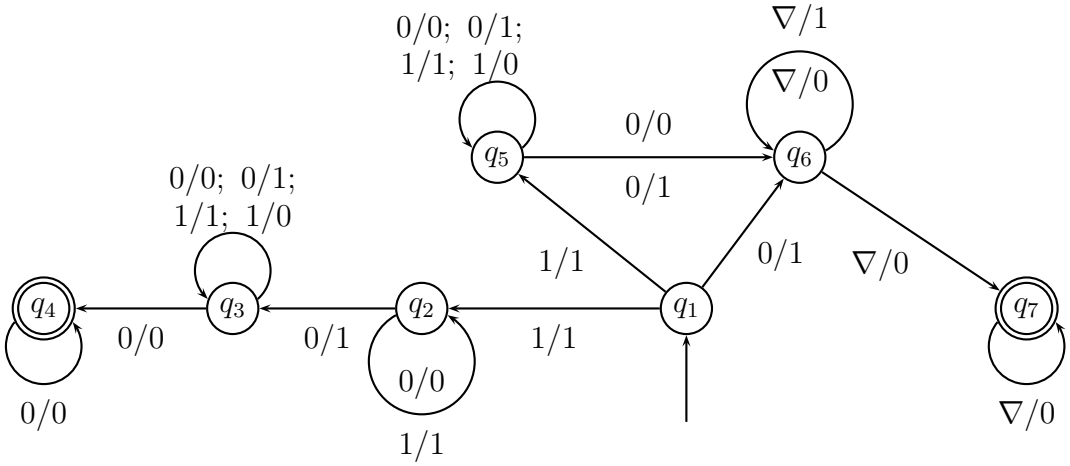
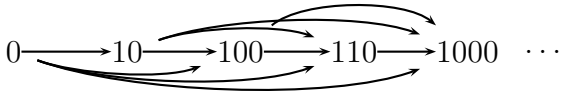


Figure 4.2: An automatic Kripke frame.



◁

The class of automatic graphs have been studied extensively under different names. Automatic relations were already studied by Büchi in 1960 under the name *sequential* relations and in 1965 by Elgot and Mezei [37] where they were called *synchronized rational* relations. Hodgson studies the satisfiability problem for first-order logic over automatic graphs in [48] and in [26]. Cannon et al. study automatic structures in order to do computations on some finitely generated groups. In [44] Frougny and Sakarovitch do a comprehensive study of the subclass of rational relations they called synchronized rational relations. This is exactly the class of regular relations. Automatic structures are reintroduced by Khoussainov and Nerode in [54] where they also define automatically presentable structures. The notion of a graph being rationally presentable is analogous to this.

Automatic structures have since been studied by various authors; see for instance Blumensath's Diploma thesis [15] where he has studied various subclasses of automatic structures. In [16, 17] automatic and ω-automatic structures have been studied and in particular the complexity of model checking and query evaluation on automatic structures for fragments of first-order logic. In his PhD. thesis, [79], Rubin studies the expressiveness and limitations of the transducers recognising automatic structures and in and [55] by Khoussainov et al. some classes of automatic structures are characterised and the isomorphism problem for automatic graphs is studied. Then automatic linear orders and also automatic trees are studied in [56] by Khoussainov et al.

4.1.1 Examples and non-examples of automatic frames

Many of the classes of infinite structures which have been studied for verification purposes are included in the class of automatic Kripke frames.

Petri nets

Let $\Sigma^{\leq k}$ denote the words in Σ^* of length less than or equal to some $k \in \mathbb{N}$ and let $LD_k = \big(\Sigma^{\leq k} \times \{\epsilon\}\big) \cup \big(\{\epsilon\} \times \Sigma^{\leq k}\big)$.

The following result can be used to determine whether a graph or a class of graphs is automatic.

Proposition 4.1.6 ([44]) *Every rational relation with bounded length difference is a regular relation.*

Proof (Sketch).[44] This follows from the fact that a rational relation with the difference between the lengths of the words it relates bounded by some k, is a finite union of products of subsets of LD_k and rational subsets of $(A \times B)^*$. QED

Example 4.1.7 In [69] Morvan states that the configuration graphs of Petri nets are rational. Moreover, Carayol and Rubin observed separately that an automatic presentation of the configuration graph of a Petri net is obtained when a unary encoding of

the markings of the Petri net is used. Let $0^{k_1}10^{k_2}1\cdots10^{k_n} \in \Sigma^*$ represent a marking $M = (k_1, \ldots, k_n)$ of a Petri net $\mathcal{N} = (P, T, F, M_0)$ where $P = \{p_1, p_2, \ldots, p_n\}$, $T = \{t_1, t_2, \ldots, t_m\}$ and $\Sigma = \{0, 1\}$. Furthermore, let R_{t_i} be the relation induced by firing the transition t_i. The value of $d_{ij} = F(t_i, p_j) - F(p_j, t_i)$ for every pair $i, j \in \{1, 2, \ldots, n\}$ is constant (Proposition 3.1.19). Hence the difference in length between two words (i.e. states) related to each other by R_{t_i} is at most $\sum_{j=1}^{n} |d_j|$ for each $i \in \{1, 2, \ldots, n\}$. From proposition 4.1.6 this relation is regular. This is true for every relation R_t with $t \in T$ and therefore the configuration graph of $\mathcal{N}$ is automatic.

For a specific example, let $\mathcal{N} = (P, T, F, M_0)$ be the Petri net with $P\{p_1, p_2\}$, $T = \{t\}$ and $M_0 = (4, 5)$. Furthermore, the flow function is defined by $F(p_1, t) = 2$; $F(t, p_2) = 3$ and $F(p_2, t) = F(t, p_1) = 0$. Then $d_{t1} = 0 - 2$ and $d_{t2} = 3 - 0$ and the difference in length between two words representing the markings and related by the relation induced by the transition t is bounded $|-2| + |3| = 5$. Since t is the only transition this is also the bound for the length difference between any two related words. In particular the word 0000100000 representing the initial marking M_0 is only related to the word 00100000000. In fact, for $\mathcal{N}$ related words will have a length difference of exactly 1. $\lhd$

A consequence of the inclusion of the configuration graphs of Petri nets in the class of automatic graphs, is that Jančar's result in [49] on the undecidability of bisimilarity for labeled Petri nets also carries over to the class of automatic Kripke frames.

Corollary 4.1.8 *It is undecidable whether two given pointed automatic Kripke models are bisimilar or not. Moreover, it is also undecidable whether two given automatic Kripke frames are globally bisimilar.*

Khoussainov et al. ([54]) prove that prefix-recognisable frames are included in the class of automatic frames. Therefore the undecidability of bisimilarity for automatic frames also follows from the undecidability of bisimilarity for prefix-recognisable frames [50].

Congruential graphs

Blumensath and Grädel give an alternative characterisation of automatic frames in [17]. This characterisation makes use of an important tool in mathematical logic, namely interpretation. A copy of a structure is defined within another structure, thus allowing the transfer of certain results among the theories. One can now use this characterisation to prove or disprove that some class of structures belongs to the class of automatic Kripke frames.

Let L be the set of first-order formulas of the natural numbers where $+$ and $=$ are the only non-logical symbols used.

Definition 4.1.9 ([76]) ***Presburger arithmetic*** *is the first-order theory of the structure* $\langle \mathbb{N}, +, = \rangle$*, i.e. the set of all sentences in* L *which are true in* $\langle \mathbb{N}, +, = \rangle$*.*

In 1929 Presburger [76] showed that Presburger Arithmetic is decidable, consistent and complete. Fisher and Rabin [43] later proved that the decision algorithm provided by Presburger has a worst case runtime of at least $2^{2^{cn}}$ for some constant $c > 0$ and where n is the length of the formula. This is one of the few examples of algorithms that provably requires more than polynomial runtime.

For convenience a number of non-logical symbols are added to the language. In fact, they serve as abbreviations for formulas in L. These non-logical symbols are the constant symbols $\mathbf{0}$ and $\mathbf{1}$, the binary relation symbol $<$ and the unary function symbol s.

The formula $\forall y \left(\mathbf{0} + y = y \right)$ now defines $\mathbf{0}$. The ordering $<$ is defined by

$$\forall x \forall y \left(\exists z \left(\neg z = \mathbf{0} \wedge x + z = y \right) \leftrightarrow x < y \right)$$

. Then $\mathbf{1}$ is defined by $\left((\mathbf{0} < \mathbf{1}) \wedge \neg \exists x \left(\mathbf{0} < x \wedge x < \mathbf{1} \right) \right)$ and s is defined by

$$\forall x \exists y \left(s\left(x\right) = y \wedge x < y \wedge \neg \exists z \left(x < z \wedge z < y \right) \right).$$

Definition 4.1.10 ([17, 15]) *A subset* $X \subseteq \mathbb{N}^n$ *is* ***Presburger definable*** *if there is a Presburger formula (first-order formula in the language of the Presburger arithmetic)* $\psi\left(\bar{x}\right)$ *with free variables* $\bar{x} = (x_1, \ldots, x_n)$ *such that* $X = \{ s \in \mathbb{N}^n | s \models \psi\left(\bar{x}\right) \}$*. Furthermore, a Kripke frame is first-order definable in* $\langle \mathbb{N}, +, = \rangle$ *if its* **domain** *and* **binary relation(s)** *are* **Presburger definable***.*

Definition 4.1.11 ([17, 15]) *A Kripke frame is* **first-order interpretable** *in* $\langle \mathbb{N}, +, = \rangle$ *if it is* **isomorphic** *to a Kripke frame that is first-order definable in* $\langle \mathbb{N}, +, = \rangle$*.*

Proposition 4.1.12 ([17]) *A Kripke frame is automatic if and only if it is first-order interpretable in the expansion* $\langle \mathbb{N}, +, |_p \rangle$ *of the Presburger arithmetic, where* $|_p : \mathbb{N} \to \mathbb{N}$ *is the relation such that* $x |_p y \iff x$ *is a power of* p *that divides* y*.*

Now consider the following class of graphs studied by Urvoy.

Definition 4.1.13 ([88]) *A* ***congruential system*** C *is a finite set of rules of the form*

$$(p, r) \xrightarrow{a} (q, s)$$

where $a \in \Gamma$ *(for* Γ *a finite alphabet of labels),* $p, q, r, s \in \mathbb{N}$*. The graph of a congruential system* C*, called a* ***congruential graph****, is given by* $\mathcal{G}\left(C\right) = \left(\mathbb{N}, R_{\mathcal{G}(C)} \right)$ *where the relation* $R_{\mathcal{G}(C)}$ *is defined by*

$$R_{\mathcal{G}(C)} = \left\{ \left(pn + r, a, qn + s \right) | \left(p, r \right) \xrightarrow{a} \left(q, s \right), n \in \mathbb{N} \right\}$$

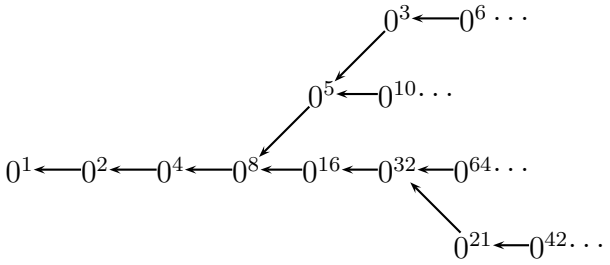
Congruential functions are presented by Conway in [34] and also by Burckel in [25]. Congruential systems are a non-deterministic generalization of these functions.

Example 4.1.14 Let C be the congruential system defined by the rules: for $n > 0$

$$\begin{aligned} 2n &\xrightarrow{a} n \\ 2n+1 &\xrightarrow{a} 3(2n+1)+1 = 6n+4 \end{aligned}$$

Then the well-known Collatz's graph [33] given in figure 4.3, is the graph of C where 0^k is short for the word $\underbrace{00\cdots0}_{k \textbf{ times}} \in \Sigma^*$ with $\Sigma = \{0\}$, and it is the unary representation of the number $k \in \mathbb{N}$.

Figure 4.3: The Collatz's graph.

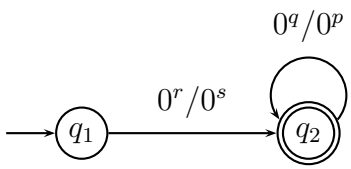


◁

Proposition 4.1.15 ([88]) *Under the unary representation of natural numbers, every congruential graph is a rational Kripke frame.*

Proof.[[88]] The relation of the graph (in unary representation) of an elementary congruential system C with the rule $nq + r \xrightarrow{a} np + s$ is recognised by the finite transducer in figure 4.4.

Figure 4.4: A finite transducer that recognises the relation of an elementary congruential graph.



Any congruential system is the finite union of elementary congruential systems and since rational relations are closed under finite union, any congruential graph is a rational Kripke frame. QED

This result can now be refined:

Proposition 4.1.16 *Congruential graphs are automatic Kripke frames.*

Proof. Consider an elementary congruential system C with the rule $nq + r \xrightarrow{a} np + s$. This rule involves only addition and multiplication with a constant which are both Presburger definable:

$$C(x,y) \iff \exists n \left(x = \underbrace{n + n + \cdots + n}_{q \text{ times}} + \bar{r} \wedge y = \underbrace{n + n + \cdots + n}_{p \text{ times}} + \bar{s} \right)$$

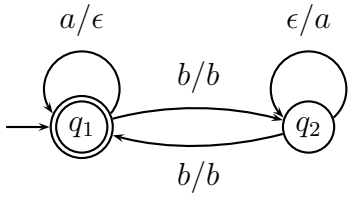
For $m \in \mathbb{N}$ let $\bar{m}$ be a constant symbol for it in the fist-order language used. Since $\bar{1}$ is Presburger definable using the ordering, so are $\bar{r}$ and $\bar{s}$. For example $\bar{r} = \underbrace{\bar{1} + \bar{1} + \cdots + \bar{1}}_{r \text{ times}}$. Only finitely many such constants will be needed since there are only finitely many rules in a system.

Then by proposition 4.1.12 the graph of this system is automatic. QED

Non-automatic rational Kripke frames

Example 4.1.17 Let $R = \{(u,v) \,|u \in \{a, bb\}^*, v \in \{ba^*b\}^*\}$. Then the transducer in figure 4.5 recognises R and R can also be represented by the regular expression $\left\{(a,\epsilon)^*, \{(b,b),(\epsilon,a)^*,(b,b)\}^*\right\}^*$. Therefore, the frame $\mathcal{F} = (S,R)$ is a rational Kripke frame.

Figure 4.5: A transducer recognising $R = \{(u,v) \,|u \in \{a, bb\}^*, v \in \{ba^*b\}^*\}$.

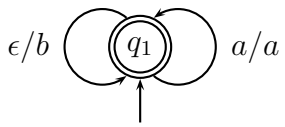


If there exists a letter-to-letter transducer that recognises the convolution of R, then it must recognise a pair of words of the type $\left(a^k (bb)^l \nabla^r, (ba^m b)^l\right)$ for $k, l, m \in \mathbb{N}$ where $(ba^m b)^l = \underbrace{ba^{m_1}bba^{m_2}b \cdot ba^{m_t}b}_{l \text{ times}}$ such that $m_1 + m_2 + \cdots + m_t - k = r$. To recognise the pair of words k edges with input label a must be read and then $2l$ edges with b as input label. The difficultly now comes in with counting the number of times a edge with a for the output label is read. This must be done in order to know how many times edges

labeled by ϵ/b must be followed to form the desired pair of words. However, automata do not have the ability to keep track of the number of times a certain type of input has been read. Push-down automata are needed for that. Hence, no such transducer exists and $\mathcal{F}$ is not an automatic Kripke frame. $\lhd$

Example 4.1.18 ([66]) Let $R = \{(A^n, u) \,|n \geq 0, u \in B^* \,(AB^*)^n\}$. Then R is a rational relation recognised by the transducer in figure 4.6 and the rational expression $\{(a, a)\,, (\epsilon, b)\}^*$ represents R. Then a Kripke frame $\mathcal{F} = (S, R)$ where $S = \{A, B\}^*$ is rational. However, by using a similar argument to the one used in Example 4.1.17, it follows that there does not exist a letter-to-letter transducer accepting the convolution of R and $\mathcal{F}$ is not automatic.

Figure 4.6: A transducer recognising $R = \{(A^n, u) \,|n \geq 0, u \in B^* \,(AB^*)^n\}$.



$\lhd$

Automatic Kripke frames of finite out-degree can be separated from the rational Kripke frames of finite out-degree by making use of the the growth rate of the out-degree of vertices.

Proposition 4.1.19 ([78]) *For any automatic frame, $\mathcal{F}$, of finite out-degree and any state s, there exists a $c \in \mathbb{N}$, such that the out-degree of vertices at distance n of x is at most c^n.*

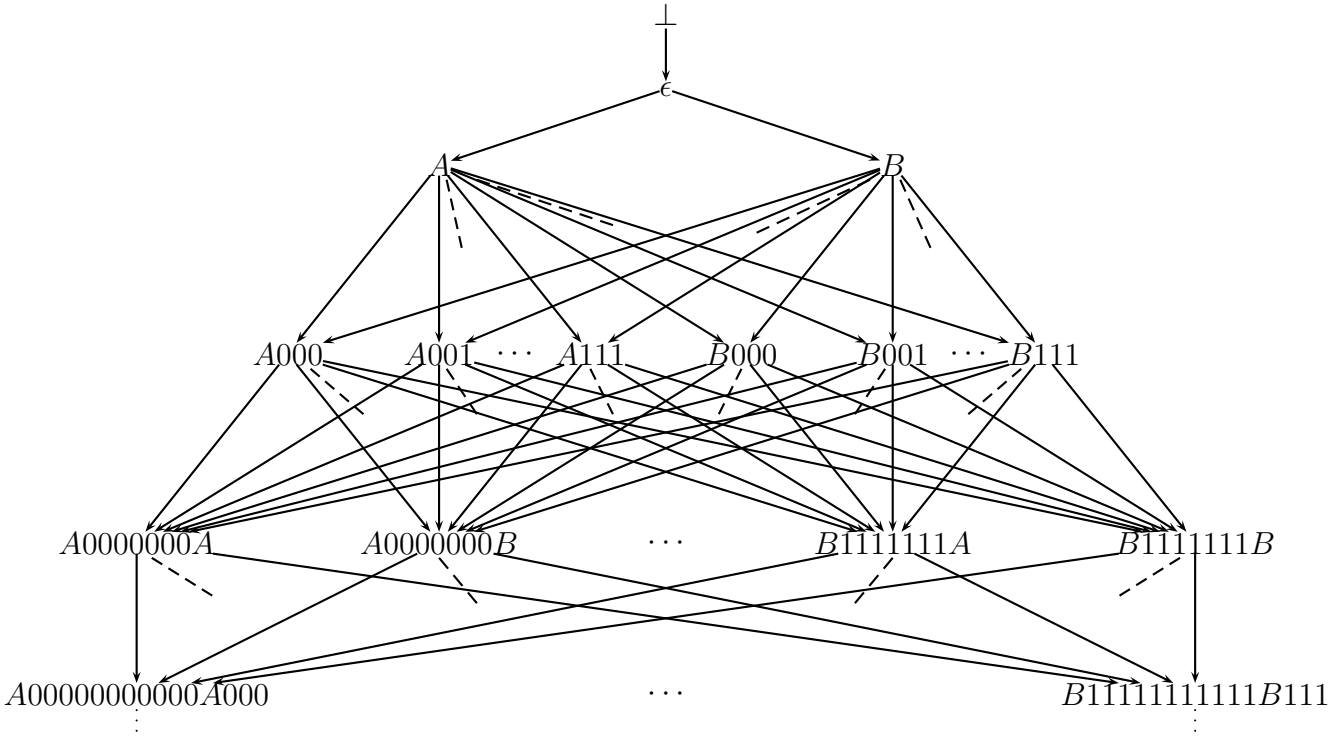
Two non-automatic rational Kripke frames are given. In the first example the growth rate of the out-degree of vertices exceeds the bound for automatic Kripke frames, but it not the growth rate of the out-degree of vertices in rational Kripke frames given in Proposition 3.1.16.

Example 4.1.20 Let $\Sigma = \{\bot, A, B, 0, 1\}$ and consider the rational Kripke frame $\mathcal{F} = (S, R)$ in figure 4.7 where the universe S is given by the regular expression $\{\{A, B\}\,\{0, 1\}^*\}^*\,\{A, b\}^* \cup \{\bot\}$ and the relation R is recognised by the transducer in figure 4.8.

At distance n from the state '$\bot$' a vertex has out-degree of 2^{n^2}. Firstly observe that a vertex at distance $n+1$ from '$\bot$', for $n \geq 1$, is a word of length n^2 and each position in the word can assume one of two letters. Moreover, the state at distance n is related to exactly $\underbrace{2 \times 2 \times \cdots \times 2}_{n^2}$ words and therefore the out-degree is 2^{n^2}.

Now, for any $c \in \mathbb{N}$ there is an $M \in \mathbb{N}$ such that $2^M \geq c$ and therefore $\left(2^M\right)^l \geq c^l$ for $n \in \mathbb{N}$. But then for all $n > M$ it holds that $n^2 > Mn$ and therefore also that $2^{n^2} > 2^{Mn} \geq c^n$. Thus there does not exist a constant c such that the out-degree of vertices at distance n from any state s is at most c^n. This proves that $\mathcal{F}$ is not an automatic frame.

Figure 4.7: A finite degree rational graph with the vertices at a distance n from any chosen vertex r having out-degree 2^{n^2}.



◁

Example 4.1.21 ([66]) Revisit the rational Kripke frame discussed in Example 3.1.17, given again in figure 4.9. This frame is not automatic since for a state s in this frame, the out-degree of states at distance n from x is $2^{2^{n+1}}$ - it exceeds the upper bound in Proposition 4.1.19.

◁

Figure 4.8: A transducer recognising the relation R.

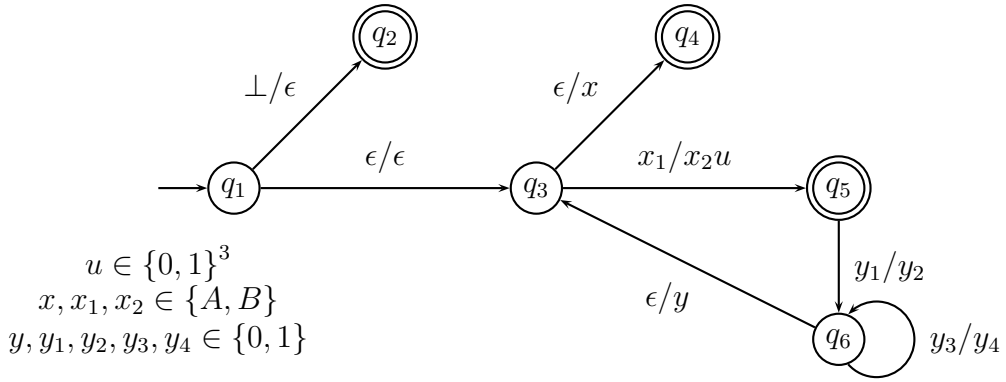
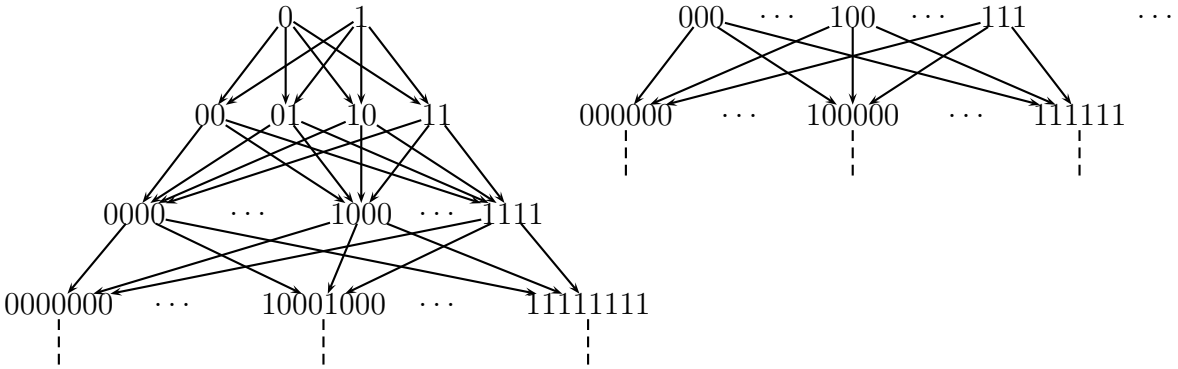


Figure 4.9: A non-automatic rational Kripke frame with finite out-degree.



4.1.2 Model checking on Automatic structures

Verification of properties for infinite state systems can be reduced to the **satisfiability problem** (see Chapter 3): the task of determining whether or not a formula with free variables is satisfiable in a given structure.

A positive result concerning the satisfiability problem for first-order logic on automatic Kripke frames has been obtained:

Theorem 4.1.22 ([48, 54]) *The satisfiability problem for first-order logic over automatic Kripke frames is decidable.*

Moreover, every automatic Kripke frame has a decidable first-order theory. This means that, fixing a frame, there is an algorithm which can determine whether a given first-order sentence is true or not.

Another problem that has been studied is **first-order query evaluation**: if an automatic Kripke frame is fixed, then, given as input a first-order formula, $\psi(\bar{x})$ with free variables $\bar{x} = (x_1, x_2, \ldots, x_n)$, determine the tuples of elements $\bar{a}$ in the frame that satisfy ψ. Formally the relation $\{\bar{a} \in S^n | \mathcal{F} \models \psi(\bar{a})\}$ is computed. First-order query evaluation for automatic frames is decidable since every such relation is an effective computable regular relation [15].

A direct consequence of these results is that model checking formula from the languages considered so far, is also decidable. The hybrid language $H_t(U, \downarrow)$ can be extended even further such that it has the same expressive power as first-order logic ([11, 13, 14]). Add the quantifier $\exists$, quantifying over the state variable, to obtain the hybrid language $H_t(U, \downarrow, \exists)$. Semantically, let $\mathcal{M} = (S, R, V)$ and $s \in S$, then:

$$\mathcal{M}, s \models \exists x \varphi \iff \text{there is a nominal } i \in \Theta \text{ such that } \mathcal{M}, s \models \varphi[x \leftarrow i]$$

Formulas in $H_t(U, \downarrow, \exists)$ can be translated to first-order formulas - see [11] for the extension of the **standard translation** [89]. Hence, local model checking of formulas in $H_t(U, \downarrow, \exists)$ is decidable. Furthermore, using this extension of the standard translation, every formula $\varphi \in H_t(U, \downarrow, \exists)$ translates into a first-order formula ψ containing exactly one free variable. The evaluation of ψ then produces a regular set. Global model checking of formulas in $H_t(U, \downarrow, \exists)$ is therefore also decidable. Moreover, for $\varphi \in H_t(U, \downarrow, \exists)$ the set $\|\varphi\|$ is regular.

Even though automatic Kripke frames enjoy positive result with respect to first-order logic, undecidability is obtained when the first-order logic is extended with reachability or transitive closure operators.

Theorem 4.1.23 ([15]) *The reachability problem over automatic graphs is undecidable.*

Therefore, even though automatic Kripke frames have decidable first-order theories, extending first-order logic with reachability gives undecidability. When decidable reachability is of interest there is no advantage to studying automatic frames over rational frames since the former does not enjoy more favourable results.

4.2 Rational Kripke trees

Definition 4.2.1 ([27]) *A **rational Kripke tree** is a rational Kripke frame that satisfies the following three properties:*

1. *The frame is connected.*

2. *Every state in the frame has at most one predecessor.*

3. *There is a unique state in the domain of the frame which is not in the range, i.e. there is a unique state in the frame which has no predecessors.*

Therefore, a **rational Kripke tree model** is a rational Kripke model with a rational Kripke tree as the underlying frame. Rational Kripke trees have been studied by Carayol and Morvan in [27].

Proposition 4.2.2 ([27]) *It is* **undecidable** *whether a rational Kripke frame is a tree.*

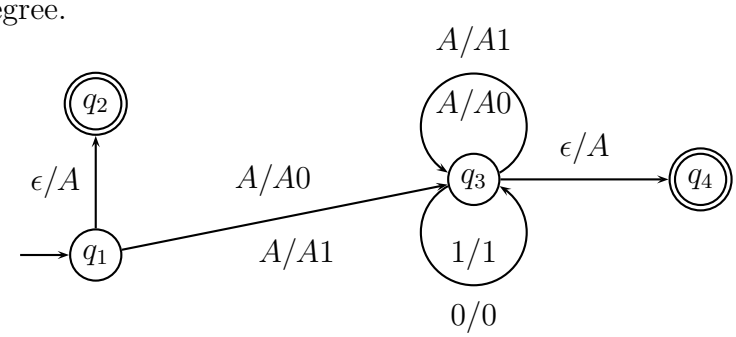
The last two of the three properties listed in the definition can be verified easily. The second property can be checked using Shutzenberger's theorem which appears in [7]. In order to verify whether a rational graph $\mathcal{G} = (S, R)$ has the third property, i.e. verifying the existence of the root, one needs to check that the set $D \backslash W$ is a singleton, where D denotes the domain of R and W its range.

However, to prove that it is undecidable whether or not a given rational graph is a rational tree, Morvan an Carayol reduces the problem to a variation of the uniform halting problem for Turing machines.

Example 4.2.3 ([27]) An example of a rational Kripke tree that is not automatic is given below.

The rational Kripke frame obtained from the finite transducer in figure 4.10 is the graph given in figure 4.11.

Figure 4.10: A finite transducer that produces a rational Kripke frame with finite, but unbounded degree.



This graph is a rational forest presented in [27] - it contains infinitely many connected components. As seen earlier in Chapter 3 on page 54, the generated subframe of a rational Kripke frame is not necessarily rational itself and therefore one can't simply claim that one of the connected components is a rational tree. From [27] it holds that given a regular language $\mathcal{L}$, the graph connecting the words in $\mathcal{L}$ into a half line in length-lexicographic order is automatic. Observe that the set of all the roots of the components in figure 4.11 form a regular language, namely $\mathcal{L}_T = A^+ \{0,1\}^* \cup \{\epsilon\}$. In

Figure 4.11: The forest such that every vertex at a depth n has 2^n successors.

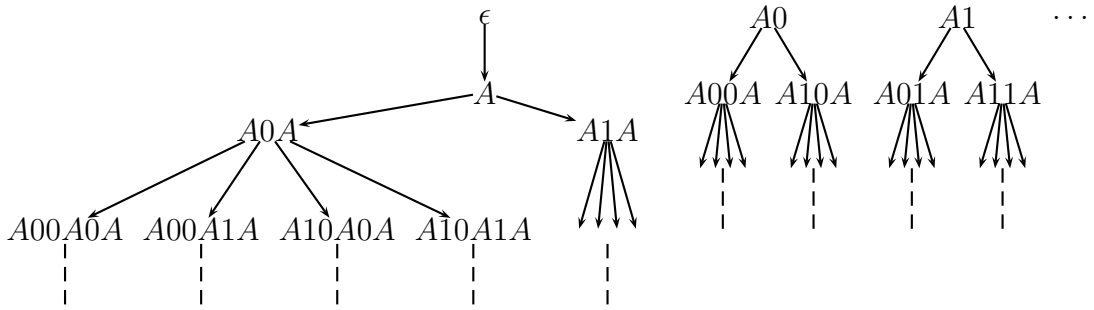


figure 4.12, I present a transducer that produces a rational graph with the words in $\mathcal{L}_T$ as vertices.

Figure 4.12: A transducer which produces a graph connecting the roots.

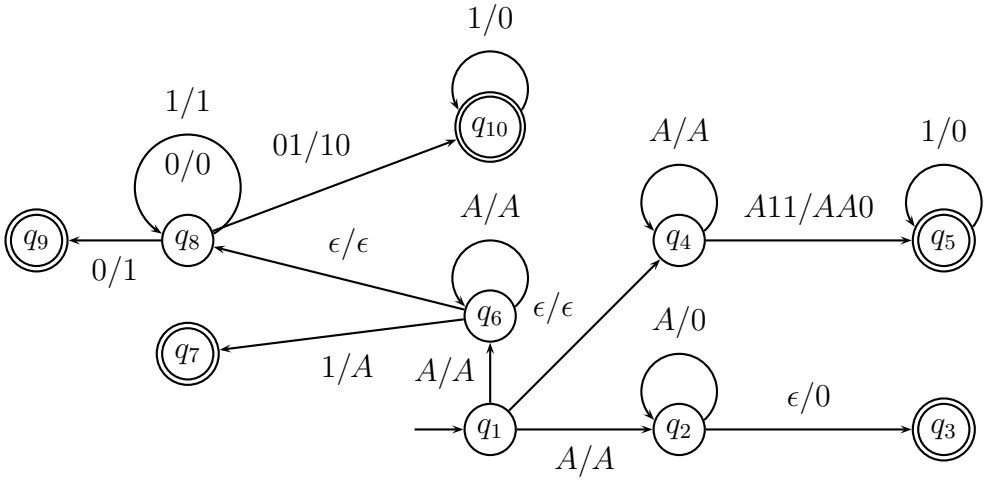


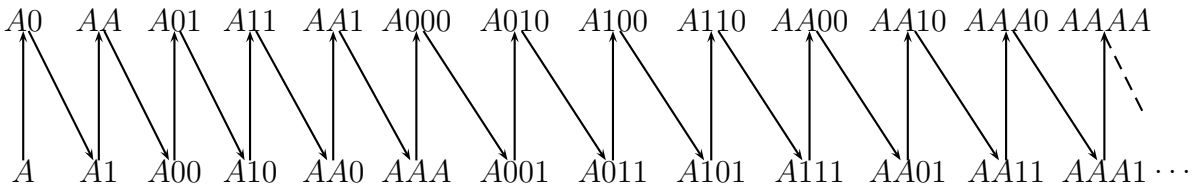
Figure 4.13 is the frame obtained from this transducer. The frames in figures 4.11 and 4.13 are both rational Kripke frames, and therefore the union of these two frames will again be rational. Moreover, it will be connected and therefore a rational Kripke tree. This tree is not automatic due to the limit of the growth rate of automatic graphs of finite degree (Proposition 4.1.19).

◁

Definition 4.2.4 ([27]) *A vertex (state) in a tree that is not the predecessor of any other state is called a* ***leaf****.*

In [71] Morvan gives the following result.

Figure 4.13: The rational graph relating all the roots of the forest.



Proposition 4.2.5 ([71]) *The set of leaves of a rational graph is a regular set.*

Proof. This result follows directly form the closure properties of regular languages. Let $\mathcal{T} = (S, R)$ be a rational tree. The domain, D, and the range, W of R are both regular. The set W will contain all the states of the tree except for the root. Thus the set $W \backslash D$ will be the set of all leaves. This set will be a regular language since regular languages are closed under complementation. QED

4.2.1 Model checking on rational Kripke trees

Proposition 4.2.6 ([71]) *Reachability is decidable for any given pair of vertices of any given rational Kripke tree.*

Proof. The decidability of the reachability follows directly from the structure of a tree: any state v is only finitely many steps away from the root. Hence the set of vertices, say W, from which v is reachable is finite. Determining whether v is reachable from u then reduces to checking whether $u \in W$. This actually checks backward reachability. QED

A language expressing reachability between two states is necessary in order to make proper use of this result.

Let $H_t\,(U, \Diamond^*)$ be the basic tense language which, in addition to being extended with nominals from the finite set Θ and the $[U]$-operator, is also extended with the reachability operator $\Diamond^*$.

Definition 4.2.7 *A formula in a hybrid language H_t is **pure** if it contains no propositional variables.*

Hence the only atomic formulas from which pure formulas are built up are the nominals.

Example 4.2.8 The pure formulas from $H_t\,(U)$ are still quite expressive.

$$\Diamond i \rightarrow \Box i$$

This formula states that if the denotation of the nominal $i \in \Theta$ is a successor of the current state, then it is the only successor of the current state.

Pure formulas from the language $H_t(U, \Diamond^*)$ can now express the reachability between two states in a model.

$$\Diamond^* i \tag{4.1}$$

$$[U](j \rightarrow \Box\Diamond^* i) \tag{4.2}$$

The first formula here says the state which is the denotation of the nominal i is reachable form the current state, while the second formula expresses: 'The denotation of i is reachable from every successor of the denotation of j.' ◁

Corollary 4.2.9 *Local model checking of pure formulas from the language $H_t(U, \Diamond^*)$ is decidable for rational Kripke trees models, i.e. given a rational Kripke tree model $\mathcal{M} = (S, R, V)$, a state $s \in S$ and a formula $\varphi \in H_t(U, \Diamond^*)$ then it is decidable whether $\mathcal{M}, s \models \varphi$ or not.*

This is a direct result of the decidability of reachability between two states.

In their study on rational graphs, Carayol and Morvan investigate the first-order theories of rational trees.

Proposition 4.2.10 ([27]) *Every rational Kripke tree has a* **decidable** *first-order theory.*

In addition Carayol and Morvan show that first-order logic with rational accessibility is undecidable for rational trees and that there exists a rational directed acyclic graph with an undecidable first-order theory.

4.3 Monotonous rational Kripke structures

The subclass of monotonous rational Kripke structures is defined by the transducers recognising their relations.

Definition 4.3.1 ([71]) *Let $\mathcal{M} = (S, R)$ be a rational Kripke frame such that the relation R is recognised by the transducer $\mathcal{T} = \langle Q, \Sigma, \Sigma, q_i, F, \rho \rangle$. If every transition $(p, u, v, q) \in \rho$ satisfies $|u| \leq |v|$ or every transition $(p, u, v, q) \in \rho$ satisfies $|u| \geq |v|$, then the rational Kripke frame is* **monotonous**.

Example 4.3.2 Consider the transducer $\mathcal{T}$ presented in figure 4.14. This is a simple example where the rational Kripke frame (see figure 4.15) generated by the transducer $\mathcal{T}$ is a monotonous rational Kripke frame.

◁

Figure 4.14: A transducer $\mathcal{T}$ producing a monotonous rational Kripke frame.

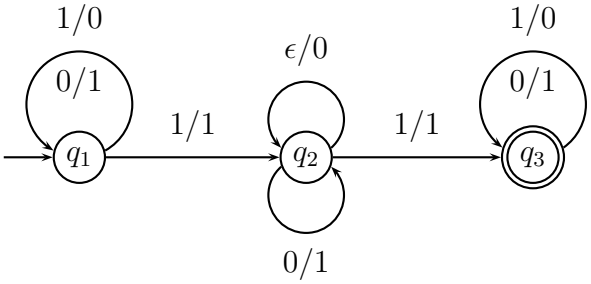
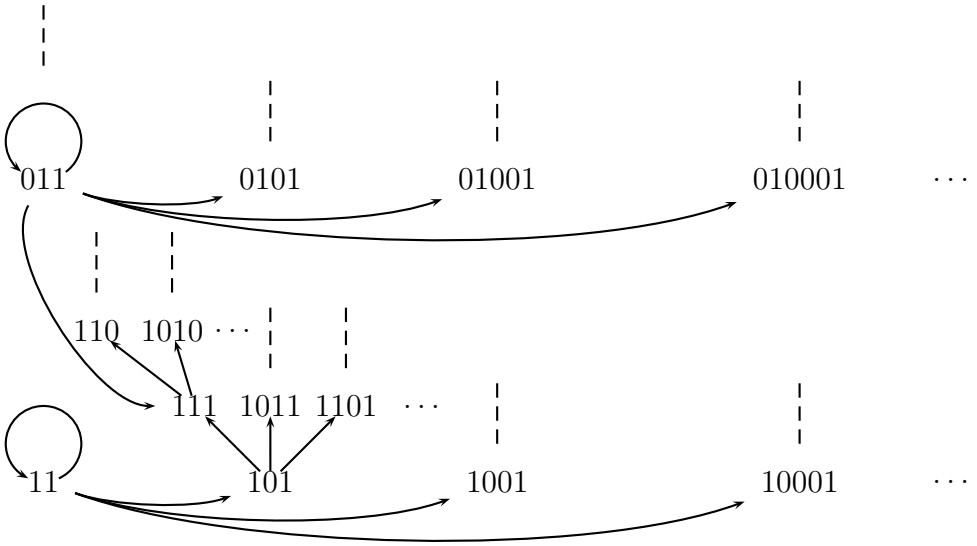


Figure 4.15: A monotonous rational Kripke frame.



Another example of a monotonous frame is the frame presented in figure 3.7 in chapter 3 - which then also serves as an example of a non-automatic monotonous frame.

4.3.1 Model checking on monotonous rational Kripke structures.

Monotonous rational Kripke frames have the following positive result with respect to reachability.

Proposition 4.3.3 ([71]) *Given a monotonous rational Kripke structure,* **reachability** *between any two given states is* **decidable**.

Proof. Fix the monotonous rational Kripke model $\mathcal{M} = (S, R, V)$ and then receive the states u and v as input. Now ask the question: 'Is the state v reachable from the state

u?' Suppose the model is upward monotonous, i.e. each transition (p, u', v', q) in the transducer recognising R satisfies $|u'| \leq |v'|$, then the subset $L = \{w : |u| \leq |w| \leq |v|\}$ of S is finite. This means there are only finitely many words w such that $|w| = |u|$ and therefore only finitely many states of the same length, $|u|$, reachable from the state u before a state with length greater than $|u|$ must be accessed. In the same way there are then only finitely many states of length $|u| + 1$ reachable from u before a state of even greater length must be accessed, etc. Thus there are finitely many possible paths from u to v. Hence the problem reduces to whether or not at least one of these finite paths belongs to the frame, which is equivalent to checking membership to a regular set a finite number of times for each path. QED

This result now makes it possible do model checking of some hybrid logic formulas on monotonous rational Kripke frames.

Corollary 4.3.4 *If φ is a pure formula in the hybrid language $H_t(U, \Diamond^*)$, then local model checking of φ on a monotonous rational Kripke frame is decidable.*

In [71] Morvan conjectures that the first-order theory of monotonous rational graphs is decidable. If this is indeed the case, then model checking of formulas from the more expressive hybrid languages on monotonous rational Kripke frames will also be decidable.

Chapter 5

An extension of rational Kripke structures

Only finite words over a finite alphabet have been considered up to now. One can also consider languages consisting of infinite strings (words). Automata on infinite words were introduced by Büchi in [23, 24]. Since then automata on infinite words and the languages they recognise have been studied by various authors, see for instance [64, 4, 62, 18, 45].

5.1 ω-rational languages

Recall that ω is the first infinite ordinal and let Σ be a finite alphabet. Then a word of infinite length, or **ω-word**, α over Σ is a sequence of letters in Σ, $\alpha = \alpha(0)\,\alpha(1)\ldots\alpha(n)\ldots$. The set of all infinite words over Σ will be denoted by Σ^ω.

ω-languages are the subsets of Σ^ω. Recall the definition of the product of two subsets $X, Y \subseteq \Sigma^*$:

$$XY = \{uv | u \in X, v \in Y\}$$

This product can now be extended to include sets containing infinite words, by setting $X \subseteq \Sigma^*$ and $Y \subseteq \Sigma^\omega$. Notice that the length of such a concatenated sequence must be the first ordinal ω. Furthermore, for $X \subseteq \Sigma^*$, let X^ω denote the set of infinite words obtained by concatenating an infinite sequence of nonempty words of X. Rational subsets of Σ^ω can now be defined.

Definition 5.1.1 ([45]) *The class of ω-**rational** subsets of Σ^ω, is the smallest set $\mathsf{RAT}_\omega(\Sigma^\omega)$ of subsets of Σ^ω such that:*

1. *if $X \in \mathsf{RAT}_\omega(\Sigma^\omega)$, then $X^\omega \in \mathsf{RAT}_\omega(\Sigma^\omega)$.*
2. *if $X \subseteq \Sigma^*$ and $Y \subseteq \Sigma^\omega$, then their product XY belongs to $\mathsf{RAT}_\omega(\Sigma^\omega)$.*

3. $\mathsf{RAT}_\omega(\Sigma^\omega)$ *is closed under finite union.*

It then follows that the **ω-rational subsets of** Σ^ω are precisely those subsets which are finite unions of sets of the form XY^ω, where X, Y are rational subsets of Σ^*.

Definition 5.1.2 ([24]) *A* ***Büchi automaton*** $\mathcal{A} = \langle Q, \Sigma, q_i, F, \delta\rangle$ *is a finite automaton, as defined in definition 2.3.2, but with the following acceptance condition: An infinite word* $\alpha \in \Sigma^\omega$ *is recognised by* $\mathcal{A}$ *if there is an infinite path*

$$(q_i, \alpha_1, q_1)(q_1, \alpha_2, q_2)\ldots(q_n, \alpha_n, q_{n+1})\ldots$$

such that $\alpha = \alpha_1\alpha_2\cdots\alpha_n\cdots$, *a finite state* $q_f \in F$ *and infinitely many* $i \in \mathbb{N}$ *such that* $q_i = q_f$.

Thus, an infinite word is recognised if a final (accepting) state is visited infinitely many times. Note that Büchi automata need not be deterministic. In fact, the class of ω-languages recognised by deterministic Büchi automata is strictly included in the class of ω-languages recognised by non-deterministic Büchi automata [64] (see also [75] for details). The language of infinite words recognised by a Büchi automaton $\mathcal{A}$ will be denoted by $\mathcal{L}_\omega(\mathcal{A})$.

In [72] Muller suggests a different acceptance condition for deterministic automaton on infinite words. For a path p, define $\mathsf{Inf}(\mathbf{p})$ to be the **set of states visited infinitely many times** by p.

Definition 5.1.3 ([72]) *A finite automaton,* $\mathcal{A} = \langle Q, \Sigma, q_i, F, \delta\rangle$ *with* $F \subseteq \mathcal{P}(Q)$, *is a* ***Muller automaton*** *if it has the acceptance condition: An infinite word* $\alpha \in \Sigma^\omega$ *is recognised by* $\mathcal{A}$ *if there is a path* $\mathsf{p} := (q_i, \alpha_1, q_1)(q_1, \alpha_2, q_2)\ldots(q_n, \alpha_n, q_{n+1})\ldots$ *such that* $\alpha = \alpha_1\alpha_2\cdots\alpha_n\cdots$ *and* $\mathsf{Inf}(\mathsf{p}) \in F$.

Note that this is a more constrained acceptance condition than Büchi's: Büchi requires that $\mathsf{Inf}(p) \cap F \neq \emptyset$ while Muller requires $\mathsf{Inf}(p) \in F$.

Instead of referring to a Büchi or Muller automaton, one can simple say an automaton with Büchi or Muller acceptance condition.

ω-rational languages can now be characterised in terms of finite machines (an analogue of Kleene's theorem for infinite words).

Theorem 5.1.4 ([64]) *A subset* X *of* Σ^ω *is* ω*-rational, if an only if*

1. $X = \mathcal{L}_\omega(\mathcal{A}_1)$ *where* $\mathcal{A}_1$ *is a* **non-deterministic** *Büchi automaton, and*

2. $X = \mathcal{L}_\omega(\mathcal{A}_2)$ *where* $\mathcal{A}_2$ *is a* **deterministic** *Muller automaton.*

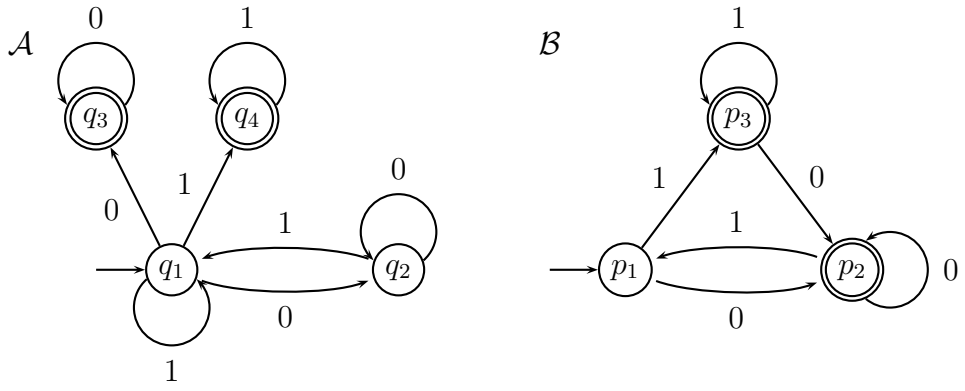
The reader is also referred to [75] for a proof of this theorem.

Example 5.1.5 Let $\Sigma = \{0, 1\}$, then the set of infinite words X_ω with at most finitely many occurrences of a '0' directly followed by an occurrence of a '1', is an ω-rational subset of Σ^ω:

$$X\left(1^*0^*01\right)^*\left(0^\omega + 1^\omega\right)$$

Figure 5.1 gives a Büchi automaton $\mathcal{A} = \langle Q_\mathcal{A}, \Sigma, q_1, \{q_3, q_4\}, \delta_\mathcal{A}\rangle$ as well as a corresponding Muller automaton $\mathcal{B} = \langle Q_\mathcal{B}, \Sigma, q_1, \{\{p_2\}, \{p_3\}\}, \delta_\mathcal{B}\rangle$ both recognising the language X_ω. Note that the Muller acceptance condition is necessary for the deterministic automaton $\mathcal{B}$ and $\mathsf{Inf}(p) \cap F \neq \emptyset$ is not enough. If $\mathcal{B}$ had only Büchi acceptance conditions then $(101)^\omega \in \mathcal{L}_\omega(\mathcal{A})$, but $(101)^\omega \notin X_\omega$.

Figure 5.1: A non-deterministic Büchi automaton and a Muller automaton both recognising $_\omega = \left(1^*0^*01\right)^*\left(0^\omega + 1^\omega\right)$.



◁

Theorem 5.1.6 *The class* $\mathsf{RAT}_\omega(\Sigma^\omega)$ *of* ω*-rational subsets is an effective Boolean algebra.*

A proof of this theorem can also be found in [75].

It is still possible to reason about finitary languages in this framework.

Lemma 5.1.7 *Let* $X \in \mathsf{REG}(\Sigma^*)$ *and suppose* $\nabla \notin \Sigma$*. The regular language* X *can be embedded in a* ω*-rational subset* X_ω *of* $(\Sigma \cup \{\nabla\})^\omega$*.*

Proof. For every $u \in X$, let $u_\omega = u\nabla^\omega$ be the associated ω-word over $\Sigma_\nabla = \Sigma \cup \{\nabla\}$. Let X_ω be the ω-language of all these words. Then there is an embedding $f : X \to X_\omega$. Moreover, since $X \in \mathsf{REG}(\Sigma^*)$ there is a finite automaton $\mathcal{A} = \langle Q, \Sigma, q^0, F, \delta\rangle$ recognising X. Let $\mathcal{A}_\omega = \left\langle Q \cup \left\{q^f\right\}, \Sigma_\nabla, q^0, \left\{q^f\right\}\delta_\omega\right\rangle$ be the Büchi automaton such that $q^f \notin Q$ and the transition relation δ_ω is obtained in the following way:

1. $\delta \subset \delta_\omega$
2. for every $q \in F$ let $\left(q, \nabla, q^f\right) \in \delta_\omega$

3. $\left(q^f, \nabla, q^f\right) \in \delta_\nabla$.

It now needs to be proven that $\mathcal{L}_\omega(\mathcal{A}) = X_\omega$:

Let $\alpha \in \mathcal{L}_\omega(\mathcal{A})$. Then there is an infinite path $(q^0, \alpha_1, q_1)(q_1, \alpha_2, q_2)\ldots(q_n, \alpha_n, q_{n+1})\ldots$ such that q^f occures infinitely often. Futhermore, there is an $i \in \mathbb{N}$ such that $\left(q_i, \nabla, q^f\right)$ and $q_j = q^f$ for all $j > i$. The path therefore has the form:

$$\left(q^0, \alpha_1, q_1\right)(q_1, \alpha_2, q_2)\ldots\left(q_i, \nabla, q^f\right)\left(q^f, \nabla, q^f\right)\ldots$$

From the construction of $\mathcal{A}_\omega$ the state q_i is a final state in $\mathcal{A}$. Thus, there is a word $u \in \mathcal{L}(\mathcal{A})$ such that $\alpha = u\nabla^\omega$ and therefore $\mathcal{L}_\omega(\mathcal{A}) \subseteq X_\omega$.

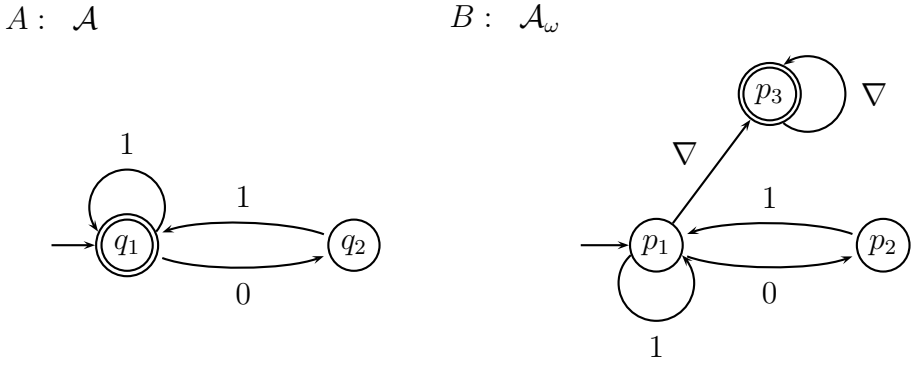
Now suppose $\alpha \in X_\omega$, then $\alpha = u\nabla^\omega$ for some $u = u_1u_2\cdots u_k \in X$. This means that there is a finite path $(q^0, u_1, q_1)(q_1, u_2, q_2)\ldots(q_k, u_k, q_{k+1})$ in $\mathcal{A}$ with $q_{k+1} \in F$. By construction the transition $\left(q_{k+1}\nabla, q^f\right) \in \delta_\omega$. Moreover, the infinite path $(q^0, u_1, q_1)(q_1, u_2, q_2)\ldots(q_k, u_k, q_{k+1})\left(q_{k+1}\nabla, q^f\right)\left(q^f, \nabla, q^f\right)\ldots$ is in $\mathcal{A}_\omega$. Hence, $\alpha \in \mathcal{L}_\omega(\mathcal{A})$ and $X_\omega \subseteq \mathcal{L}_\omega(\mathcal{A})$.

QED

Example 5.1.8 This example illustrates the embedding of a regular language in an ω-rational language. Let $X \in \mathsf{REG}(\Sigma^*)$ be the set of all finite words over $\Sigma = \{0, 1\}$ in which every occurrence of a '0' is directly followed by an occurrence of a '1'. Then $X = (1 + 01)^*$ and X is recognised by the finite automaton $\mathcal{A}$ in figure 5.2.

Now let $\Sigma_\nabla = \{0, 1, \nabla\}$, then X is embedded into the ω-rational language $X_\omega = (1 + 01)^*\nabla^\omega$ which is recognised by the Büchi automaton $\mathcal{A}_\omega$ in figure 5.2.

Figure 5.2: The finite automaton $\mathcal{A}$ and the Büchi automaton $\mathcal{A}_\omega$.



◁

ω-rational languages can therefore be seen as a generalisation of finitary regular languages.

5.2 ω-rational relations

Rational relations on infinite words, called **ω-rational relations** can now be defined. They have been studied by various authors: Barzdin and Trakhtenbrot [4], Lindner and Staigner [62], Nivat and Gire [45] and later also by Frougny and Sakarovitch [44].

Definition 5.2.1 ([42]) *An* ***ω-rational relation*** *over a finite alphabet* Σ*, is a subset of* $\Sigma^\omega \times \Sigma^\omega$ *which are the finite union of relations of the form* $R_1 R_2^\omega$ *where* R_1 *and* R_2 *are rational relations over* $\Sigma^* \times \Sigma^*$ *.*

Analogous to the case of ω-languages, the finite machines recognising rational relations on finite words can be modified to obtain machines recognising relations on infinite words.

Definition 5.2.2 ([24, 5]) *A finite transducer* $\mathcal{T} = \langle Q, \Sigma, \Gamma, q^0, F, \rho \rangle$ *is a* ***Büchi transducer*** *if it has the following acceptance condition:*

A pair $(\alpha, \beta) \in \Sigma^\omega \times \Gamma^\omega$ *is recognised by* $\mathcal{T}$*, if for* $\alpha = \alpha_1 \alpha_2 \ldots \alpha_n \ldots$ *and* $\beta = \beta_1 \beta_2 \ldots \beta_n \ldots$ *there is an infinite path*

$$(q_i, \alpha_1, \beta_1, p_1)\,(p_1, \alpha_2, \beta_2, p_2) \ldots (q_n, \alpha_{n+1}, \beta_{n+1}, p_{n+1}) \ldots$$

and a final state $q_f \in F$ *such that* $q_f = p_n$ *for infinitely many non-negative integers* n*.*

Since α and β are ω-words, only finitely many of the α_i's and β_i's may be the empty word, i.e. there exist $k, l \in \mathbb{N}$ such that $\alpha_j \neq \epsilon$ for every $j > k$ and $\beta_j \neq \epsilon$ for every $j > l$.

Definition 5.2.3 ([44]) *A transducer* $\mathcal{T} = \langle Q, \Sigma, \Gamma, q^0, F, \rho \rangle$ *is* **deterministic** *if it satisfies the following:*

1. *There is a partitioning* $Q = Q_1 \cup Q_2$ *such that*
   - *(a) for every* $q \in Q_1$*, if* $(q, u, v, p) \in \rho$ *then* $u \in \Sigma$ *and* $v = \epsilon$*.*
   - *(b) for every* $q \in Q_2$*, if* $(q, u, v, p) \in \rho$ *then* $u = \epsilon$ *and* $v \in \Gamma$*.*
2. q^0 *is unique, i.e the only initial state.*
3. *For every input (respectively, output) symbol at most a single transition to a new state is possible.*

The transducer $\mathcal{T}$ *is* **non-deterministic** *whether it satisfies these conditions or not.*

Theorem 5.2.4 ([41, 5]) *A relation* R *over* $\Sigma^\omega \times \Sigma^\omega$ *is rational if and only if it is recognised by a non-deterministic Büchi transducer.*

In fact, in the literature ω-rational relations are usually defined as the class of languages accepted by Büchi transducers and not in terms of ω-rational expressions.

As with automata, the Muller acceptance condition can be used instead of the Büchi condition.

Definition 5.2.5 ([72, 5]) *A finite transducer $\mathcal{T} = \langle Q, \Sigma, \Gamma, q^0, F, \rho\rangle$ is a **Muller transducer** if $F \subseteq \mathcal{P}(Q)$ and it has the following acceptance conditions:*

A pair $(\alpha, \beta) \in \Sigma^\omega \times \Gamma^\omega$ is recognised by $\mathcal{T}$, if for $\alpha = \alpha_1\alpha_2 \ldots \alpha_n \ldots$ and $\beta = \beta_1\beta_2 \ldots \beta_n \ldots$ there is an infinite path

$$\mathsf{p} := (q_i, \alpha_1, \beta_1, p_1)(p_1, \alpha_2, \beta_2, p_2) \ldots (q_n, \alpha_{n+1}, \beta_{n+1}, p_{n+1}) \ldots$$

such that $\mathsf{Inf}(\mathsf{p}) \in F$.

Lemma 5.2.6 ([5]) *An ω-relation is accepted by a nondeterministic Büchi transducer if and only if it is accepted by a non-deterministic Muller transducer.*

Proof (Sketch).([5]) If a Büchi transducer $\mathcal{T}_B = \langle Q, \Sigma, \Gamma, q^0, F_B, \rho\rangle$ then $\mathcal{T}_M = \langle Q, \Sigma, \Gamma, q^0, F_M, \rho\rangle$ where $F_M = \{P \subseteq Q | P \cap F_B \neq \emptyset\}$, is a Muller transducer which recognises the relation as $\mathcal{T}_B$.

In [85] Thomas gives a construction of a Büchi automaton from a given Muller automaton such that the two automata recognise the same ω-rational language. This construction can now be modified slightly to produce a Büchi transducer recognising the same relation as a given Muller transducer. QED

This lemma states that non-deterministic Büchi and Muller transducers have the same accepting power and that ω-rational relations are exactly those accepted by a non-deterministic Muller transducers.

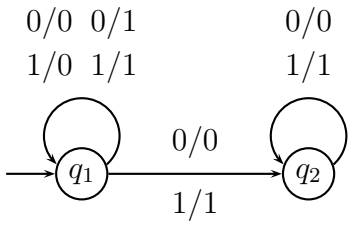
Example 5.2.7 Let R be the ω-rational relation such that two ω-words are related by R if and only if there are finitely many corresponding symbols in which they differ, i.e. $R = ((0,0) + (0,1) + (1,0) + (1,1))^* ((0,0) + (1,1))^\omega$ with $F = \{q_2\}$. If the transducer $\mathcal{T}$ given in figure 5.3 is equipped with Büchi acceptance conditions with $F = \{q_2\}$ being the set of final states, then $\mathcal{T}$ recognises R.

By the construction given in [5] mentioned in the proof of Lemma 5.2.6, R is also accepted by $\mathcal{T}$ under Muller acceptance conditions if $F = \{\{q_1, q_2\}, \{q_2\}\}$. Since q_1 has no incoming edges, no pair of ω-words are accepted by $\mathcal{T}$ because of an infinite path p for which $\mathsf{Inf}(\mathsf{p}) = \{q_1, q_2\}$. Hence, only if $\mathsf{Inf}(\mathsf{p}) = \{q_2\}$ does the path form an accepting run.

$\lhd$

Rational relations over finite words can also be embedded into ω-rational relations.

Figure 5.3: A transducer $\mathcal{T}$ which can be equipped with either Büchi or Muller acceptance conditions in order for it to recognise the ω-rational relation R.



Lemma 5.2.8 *A rational relation $R \subseteq \Sigma^* \times \Sigma^*$ can be embedded into an ω-rational relation $R_\omega \subseteq \Sigma_\nabla^\omega \times \Sigma_\nabla^\omega$ where $\Sigma_\nabla = \Sigma \cup \{\nabla\}$.*

The proof of this lemma is analogous to the proof of lemma 5.1.7 and the idea is illustrated in the example below.

Example 5.2.9 The relation $R = ((0,1)(1,1)^*(0,1))^*$ is recognised by the transducer in figure 5.4 and can be embedded into the relation $R_\omega = ((0,1)(1,1)^*(0,1))^*(\nabla,\nabla)^\omega$ which is recognised by the transducer in figure 5.5

Figure 5.4: A transducer $\mathcal{T}$ recognising $R = ((0,1)(1,1)^*(0,1))^*$.

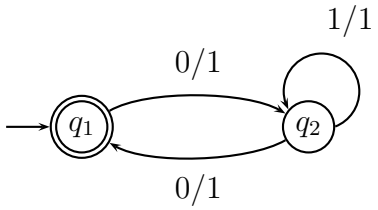
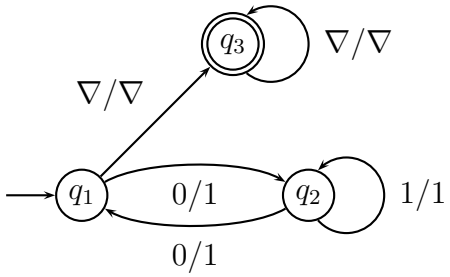


Figure 5.5: A transducer $\mathcal{T}_\omega$ recognising R_ω.



◁

5.3 ω-rational Kripke structures

Finally, rational Kripke frames (models) can be extended to ω-rational Kripke frames (models).

Definition 5.3.1 *An* ***ω-rational Kripke frame*** *is a Kripke frame $\langle S, R\rangle$ where S is an ω-rational subset of Σ^ω, for a finite alphabet Σ and $R \subseteq \Sigma^\omega \times \Sigma^\omega$ is an ω-rational relation over Σ.*

Definition 5.3.2 *An* ***ω-rational Kripke model*** *is a Kripke model $\langle \mathcal{F}, V\rangle$, based on an ω-rational Kripke frame, equipped with an* **ω-rational valuation**, *i.e. V assigns an ω-rational subset of Σ^ω to every propositional variable $p \in \Phi$.*

Note that the class of rational Kripke frames can be embedded into the class of ω-rational Kripke frames. Likewise for ω-rational Kripke models. This is done by embedding the universe and the relation of the structure into an ω-rational language and an ω-rational relation over the same extended alphabet. In the case of models, the regular languages assigned to the propositional variables by the valuation are also embedded into ω-rational languages over this extended alphabet.

Proposition 5.3.3 *The classes of ω-rational Kripke structures are finitely presentable.*

Proof. The set of states and the valuation can be presented by finitely many Büchi or Muller automata while a Büchi or a Muller transducer can be used to represent the relation. QED

Example 5.3.4 Let $\mathcal{M} = \langle S_\omega, R_\omega, V_\omega\rangle$ be an ω-rational Kripke model, and let the set of propositional variables be $\Phi = \{p, q\}$. Furthermore, let the set $S_\omega = \Sigma^\omega = \{0, 1\}^\omega$ and let R_ω be the ω-rational relation, such that R_ω relates a word α to a word β if α and β share a finite prefix. Finally let $V_\omega(p)$ be the ω-rational language with no more than one occurrence of '0' and $V_\omega(q)$ contain the ω-words in which every letter in Σ occur at least once. The model $\mathcal{M}$ can then be finitely presented by the various machines with Büchi acceptance conditions in figure 5.6.

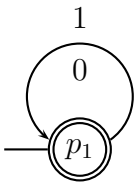
$\lhd$

Any rational Kripke structure has countably many states since the states are labeled by a regular language. This is however not the case for ω-rational Kripke structures. Their state spaces are labeled by ω-rational languages and can therefore be uncountable.

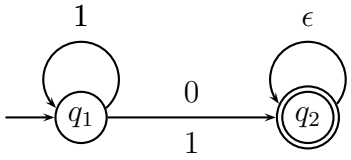
Thus, in order to show that ω-rational Kripke frames are indeed a generalisation of rational Kripke frames I will now give an example of an ω-rational Kripke frame which is not rationally presentable.

Figure 5.6: Büchi automata and transducers representing the ω-rational Kripke model $\mathcal{M}$.

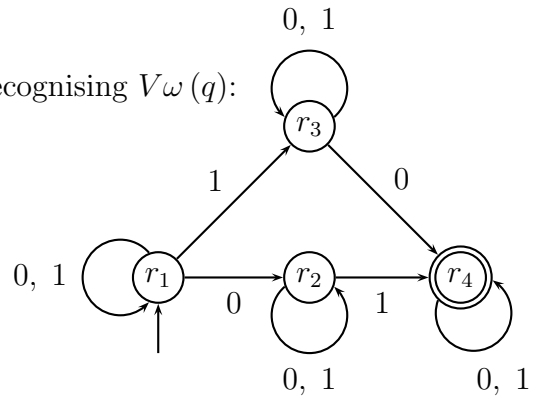
$\mathcal{A}_1$ recognising S_ω:



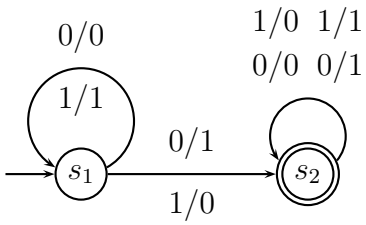
$\mathcal{A}_2$ recognising $V_\omega(p)$:



$\mathcal{A}_3$ recognising $V\omega(q)$:



$\mathcal{T}$ recognising R_ω:



Definition 5.3.5 *The* **characteristic string** *of a subset X of $\mathbb{N}$ is an ω-word α over the alphabet $\Sigma = \{0,1\}$ such that $\alpha(i) = 1$ if $i \in X$ for every $i \in \mathbb{N}$ where $\alpha(i)$ is the i^{th} character in α. The characteristic string associated with a set X is denoted by $\mathbf{C_X}$.*

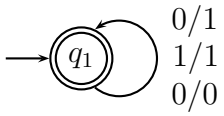
Example 5.3.6 Let $\mathcal{F} = \langle \mathcal{P}(\mathbb{N}), \subseteq \rangle$ be a Kripke frame with the set of subsets of the natural numbers as its universe and set inclusion as its relation. Clearly the universe is an ω-rational set of Σ^ω - each subset is simply presented by its characteristic string. In fact Σ^ω represents the universe.

Now let $X, Y \in \mathcal{P}(\mathbb{N})$ then $X \subseteq Y$ if and only if C_X only has a '1' in position i if C_Y has a '1' in position i. But this is exactly the relation recognised by the Büchi transducer in figure 5.7.

Hence, $\mathcal{F}$ is an ω-rational Kripke frame. ◁

Since $\mathcal{P}(\mathbb{N})$ is uncountably infinite, it cannot be labeled by a regular language and cannot be the universe of a rational Kripke frame.

Figure 5.7: The Büchi transducer recognising the relation $\{(C_X, C_Y) \,|X \subseteq Y\}$.



5.4 Model checking on ω-rational Kripke models

In this section I investigate model checking K_t on ω-rational Kripke models.

Question 5.4.1 *Is model checking of K_t-formulas decidable on ω-rational Kripke models?*

I will provide a solution similar to the case of finite words, i.e a synchronised product of transducers and automata over infinite words. It must be decided which acceptance conditions to use and how to define the acceptance conditions of the synchronised product.

One may easily be tempted to alter the definition of the synchronised product of a finite automaton and a finite transducer by just adding Büchi acceptance conditions. From the example below it is clear that this is not sufficient and that more care must be taken with the acceptance conditions.

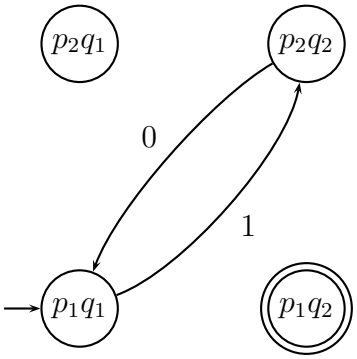
Example 5.4.2 Let $\mathcal{T}$ and $\mathcal{A}$, given in figure 5.8 be a Büchi transducer and a Büchi automaton recognising R_ω and X_ω respectively. Let $\mathcal{B}$, see figure 5.9, be the Büchi automaton obtained from the synchronisation of their transition relations as well as the cross product of their sets of final states. Then $\mathcal{B}$ does not accept any ω-word and consequently does not recognise $\langle R_\omega \rangle X_\omega$.

Figure 5.8: The Büchi automaton $\mathcal{A}$ and the Büchi transducer $\mathcal{T}$.

◁

Definition 5.4.3 *Let $\mathcal{T}_\omega = \langle Q_{\mathcal{T}_\omega}, \Sigma, q^0_{\mathcal{T}_\omega}, F_{\mathcal{T}_\omega}, \rho_{\mathcal{T}_\omega} \rangle$ be a Muller transducer, and let $\mathcal{A}_\omega = \langle Q_{\mathcal{A}_\omega}, \Sigma, q^0_{\mathcal{A}_\omega}, F_{\mathcal{A}_\omega}, \delta_{\mathcal{A}_\omega} \rangle$ be a Muller automaton.*

Figure 5.9: A Büchi automaton $\mathcal{B}$.



*The **synchronised product** of $\mathcal{T}_\omega$ with $\mathcal{A}_\omega$ is the Muller automaton:*

$$\mathcal{T}_\omega \prec \mathcal{A}_\omega = \left\langle Q_{\mathcal{T}_\omega} \times Q_{\mathcal{A}_\omega}, \Sigma, \left(q^0_{\mathcal{T}_\omega}, q^0_{\mathcal{A}_\omega}\right), F_{\mathcal{T}_\omega \prec \mathcal{A}_\omega}, \delta_{\mathcal{T}_\omega \prec \mathcal{A}_\omega} \right\rangle$$

where $\delta_{\mathcal{T}_\omega \prec \mathcal{A}_\omega} : (Q_{\mathcal{T}_\omega} \times Q_{\mathcal{A}_\omega}) \times (\Sigma \cup \{\epsilon\}) \to \mathcal{P}(Q_{\mathcal{T}_\omega} \times Q_{\mathcal{A}_\omega})$ is defined in the same way $\delta_{\mathcal{T} \prec \mathcal{A}}$ was definined for a finite transducer $\mathcal{T}$ and a finite automaton $\mathcal{A}$. See Definition 3.3.5 for the details.

Futhermore, $F_{\mathcal{T}_\omega \prec \mathcal{A}_\omega}$ is defined as follows: Let $S_i^{\mathcal{T}} \in F_{\mathcal{T}_\omega}$ and $S_j^{\mathcal{A}} \in F_{\mathcal{A}_\omega}$. Then $S_{ij} = S_i^{\mathcal{T}} \times S_j^{\mathcal{A}}$. Now set

$$\mathsf{S}_{ij} = \mathcal{P}\left(S_{ij}\right)$$

and

$$\underline{\mathsf{S}_{ij}} = \left\{ Y \in \mathsf{S} | \left(\exists q \in F_{\mathcal{T}_\omega} \forall \left(r_1, r_2\right) \in Y \left(r_1 \neq q\right)\right) \vee \left(\exists q \in F_{\mathcal{A}_\omega} \forall \left(r_1, r_2\right) \in Y \left(r_2 \neq q\right)\right) \right\}$$

Finally set $F_{ij} = \mathsf{S}_{ij} \backslash \underline{\mathsf{S}_{ij}}$. Then $F_{\mathcal{T}_\omega \prec \mathcal{A}_\omega} = \bigcup_{S_i^{\mathcal{T}} \in F_{\mathcal{T}_\omega},\, S_j^{\mathcal{A}} \in F_{\mathcal{A}_\omega}} F_{ij}$

Again, the synchronization of accepting runs of $\mathcal{T}_\omega$ and $\mathcal{A}_\omega$ is an accepting run of $R_{\mathcal{T} \prec \mathcal{A}}$.

Lemma 5.4.4 *Let $\mathcal{T}_\omega = \left\langle Q_{\mathcal{T}_\omega}, \Sigma, q^0_{\mathcal{T}_\omega}, F_{\mathcal{T}_\omega}, \rho_{\mathcal{T}_\omega} \right\rangle$ be a Muller transducer recognizing the ω-rational relation $R(\mathcal{T}_\omega)$ and let $\mathcal{A}_\omega = \left\langle Q_{\mathcal{A}_\omega}, \Sigma, q^0_{\mathcal{A}_\omega}, F_{\mathcal{A}_\omega}, \delta_{\mathcal{A}_\omega} \right\rangle$ be a Muller automaton recognizing the ω-rational language $L(\mathcal{A}_\omega)$. Then the ω-rational language recognized by the synchronised product of $\mathcal{T}_\omega$ and $\mathcal{A}_\omega$ is*

$$L(\mathcal{T}_\omega \prec \mathcal{A}_\omega) = \{\mathbf{u} \mid \exists \mathbf{w} \in L^\epsilon(\mathcal{A}_\omega)(\mathbf{u} R(\mathcal{T}_\omega) \mathbf{w})\}.$$

Lemma 5.4.5 *Let Σ be a finite non-empty alphabet, $X_\omega \subseteq \Sigma^\omega$ an ω-rational subset, and let $R_\omega \subseteq \Sigma^\omega \times \Sigma^\omega$ be an ω-rational relation. Then the sets*

$$\langle R_\omega \rangle X_\omega = \{u \in \Sigma^\omega | \exists v \in X_\omega \left(u R_\omega v\right)\}$$

and

$$\left\langle R_\omega^{-1} \right\rangle X_\omega = \{u \in \Sigma^\omega | \exists v \in X_\omega \, (vR_\omega u)\}$$

are ω-rational subsets of Σ^ω.

Proof. $\langle R_\omega \rangle X_\omega$ is the ϵ-reduction of the ω-rational language recognised by the synchronised product of a Muller transducer $\mathcal{T}_\omega$ recognising R_ω and a Muller automaton $\mathcal{A}_\omega$ recognising X_ω. QED

Example 5.4.6 Suppose $\mathcal{T}_\omega$ and $\mathcal{A}_\omega$ are the transducer and automaton in Example 5.4.2 respectively, but both with the Muller acceptance condition for $F_{\mathcal{T}_\omega} = \{p_1, p_2\}$ and $F_{\mathcal{A}_\omega} = \{q_1, q_2\}$. Let R_ω be the ω-rational relation recognised by $\mathcal{T}_\omega$ and X_ω the ω-rational relation recognised by $\mathcal{A}_\omega$. Then $\{(p_1, q_1), (p_2, q_2)\} \in F_{\mathcal{T}_\omega \nearrow \mathcal{A}_\omega}$, and the synchronised product of $\mathcal{T}_\omega$ and $\mathcal{A}_\omega$ recognise $\langle R_\omega \rangle X_\omega$. $\lhd$

Global model checking of a formula in an ω-rational Kripke model, is defined as before: it produces the set of states, which satisfies the formula, symbolically. In the case of local model checking a formula, one needs to be more careful since the states are encoded as ω-sequences. There are uncountably many ω-sequences, while only countably many presentation. Therefore, local model checking a formula on an ω-rational Kripke model will be restricted to those states which can be represented by a Büchi-automaton, i.e. local model checking can be done at a state s if the set $\{s\}$ forms an ω-rational language.

Theorem 5.4.7 *Local and global model checking of formulas in K_t on ω-rational Kripke models are decidable. Moreover, for every basic modal formula φ and every ω-rational Kripke model $\mathcal{M}$, the set $\|\varphi\|_{\mathcal{M}}$ is an effectively computable ω-rational language.*

Theorem 5.4.8 *Local and global model checking of formulas from the hybrid language $H_t(U)$ is decidable on ω-rational Kripke models and as before the set $\|\varphi\|_{\mathcal{M}}$ is an ω-rational language for every formula $\varphi \in H_t(U)$ and every ω-rational model $\mathcal{M}$.*

The proofs of these two results are analogous to proofs for rational Kripke models. The claims follow from the closure properties of ω-rational languages as well as Lemma 5.4.5.

It turns out that no expressiveness in terms of decidable model checking is sacrificed with the extension of rational Kripke structures to ω-rational Kripke structures - therefore justifying the study of ω-rational Kripke structures.

Chapter 6

Conclusion

6.1 Summary

In this dissertation I have studied Rational Kripke models - a study motivated by increasing interest in model checking in infinite structures.

In particular, it has been shown that global and local model checking of formulas of the basic tense logic K_t (and the hybrid extension $H_t(U)$ thereof) on rational Kripke models are decidable. Moreover, a constructive proof is presented where an automaton recognising the regular language $\langle R \rangle X$ is given explicitly.

The class of regularity preserving relations has been studied and two new types of two-sorted algebras have be defined: regularity preserving- and rational-Boolean algebras. The classes of regularity preserving and rational relations then respectively form algebras of the two types.

Model theoretic results for rational Kripke models have been presented and in particular bisimulation equivalence between rational Kripke models has been studied. It is shown that bisimilarity between a given pointed rational Kripke modal and a given finite pointed Kripke model is decidable.

I have also studied three subclasses of rational Kripke structures and summarised some of the existing results on these structures. The hybrid language $H_t(U, \Diamond^*)$ turns out to be useful in expressing reachability between two given states, and as a result model checking pure formulas of $H_t(U, \Diamond^*)$ is decidable on rational Kripke tree models and monotonous rational Kripke models.

Finally, rational Kripke models have been extended to ω-rational Kripke models. This extension is justified by the decidability of model checking formulas of K_t in ω-rational Kripke models.

6.2 Applications

I will now briefly discuss two potential applications of the model checking modal logic to the formal verification of infinite state systems.

6.2.1 Regular model checking

A potential application is regular model checking [73]. This is a framework for algorithmic verification of generally infinite state systems which essentially involves computing reachability sets in regular Kripke models. In [20] Bouajjani et al. present two complementary techniques for solving two major problems in verification, namely computing the set of states reachable from a set of initial states, and computing the transitive closure of the transition relation.

The first technique is direct automata-theoretic construction. Here the *column transducer* recognising the transitive closure of a length-preserving regular relation R is constructed. The column transducer is however not finite in general. In [52] Jonsson and Nilsson gave some sufficient conditions under which the transitive closure of a length-preserving regular relation is itself regular. For these relations the column transducer will be finite. These are only sufficient conditions for the transitive closure to be regular and there are many other relations for which the column transducer is also finite. Since the transitive closure of a relation can be used to compute the set of reachable states, this serves as a solution to both problems mentioned.

The second technique discussed in [20] involves widening on regular sets - these are techniques to compute the effect of iterating a length-preserving regular relation. The idea is to guess automatically what the image will be when iterating a relation and starting with a given regular set, and then to decide whether the guess is correct. The guessing is done by comparing the given starting regular set X with the regular set $R(X)$. A fixpoint test is used to ensure that an upper approximation of the set of reachable states is guessed.

These techniques can now be generalised for rational relations instead of length-preserving regular relations. In the first case the only difference would be that since the relation R will now be rational, the transducer accepting it will be rational and therefore will accept ϵ-transitions. This does not affect the construction of the column transducer. Similarly, the widening on regular sets can be applied to rational relations.

6.2.2 Bounded model checking

In [57] Schneider et al. describe a forerunner of bounded model checking. Bounded model checking has been introduced in [9] by Biere et al, and has since been studied extensively, see for instance [8], as an alternative to symbolic model checking techniques where binary decision diagrams (data structures used to represent Boolean functions)

are used [1, 60, 68]. Most propositional binary decision diagrams suffer from an unavoidable exponential blow-up [65]. Consequently model checkers based on binary decision diagrams may fail on relatively small models. Bounded model checking was introduced in an attempt to deal with cases like these.

The basic idea behind bounded model checking is to reduce the model checking task to the satisfiability problem of some base logic. The base logic is usually propositional logic and the model checking task is usually reduced to the Boolean satisfiability problem (SAT). In technical terms, the specification to be model checked (expressed in temporal logic) is unwound a finite number of times to obtain a formula in the base logic. Therefore bounded model checking is mainly used for the verification or falsification of **liveness** and **safety** properties respectively.

Safety properties are specifications that hold invariantly along a given path whereas liveness properties are specifications which must hold at least once along a given path. Bounded model checking is used to produce counterexamples for safety properties or witnesses for liveness properties. Properties like **liveness** and **safety** can be expressed in linear- and branching-time temporal logics.

Example 6.2.1 The linear-time temporal formula $\varphi = Fp$ in $\mathrm{LTL}_{\mathrm{past}}$ specifies a liveness property: eventually there must be a state that satisfies the atomic proposition p.

The branching-time formula $\mathbf{A}\varphi$ in CTL^* specifies that the liveness property must hold along every path (starting at the current state).

The formula $\psi = G\,(p \vee q)$ in $\mathrm{LTL}_{\mathrm{past}}$ specifies a safety property: at least one of p and q must always be satisfied.

Then $\mathbf{E}\psi \in \mathrm{CTL}^*$ states that there must be a path starting at the current state such that at least one of p and q is satisfied at every state in the path. $\lhd$

Model checking of these properties on models with infinite state-spaces is therefore undecidable. It then follows that bounded model checking is an incomplete infinite-state verification method.

Moreover, even though some problems can be solved by bounded model checking but cannot be solved by BDD-base model checking techniques, the converse is also true. Some problems can be solved better by BDD-based techniques. Bounded model checking is therefore an automatic verification method that supplements other existing methods but does not replace any of them [8]. Furthermore, since bounded model checking relies on an exponential procedure it does not solve the complexity problem of model checking. Some interesting problems can however be solved faster by bounded model checking.

In [35] the infinite state models (transition systems) on which model checking is done must be expressible in terms of a set of Boolean constraints. In [82, 81] integer Kripke models, where the relation and assignment must be expressible expressible in

Presburger arithmetic, are used. Consequently, the bounded model checking techniques were studied with rather restricted classes of infinite structures in mind.

Alternatively, the class of rational Kripke models is a very general class of (possibly) infinite structures on which model checking the basic tense logic K_t is decidable. Rational Kripke models are thus suitable candidates for bounded model checking of temporal logic formulas with K_t as the base logic.

6.3 Open problems

As I have said before, rational graphs have not been studied very widely in the literature. Consequently, there are still many unanswered questions concerning rational Kripke structures.

Firstly, an open problem is given in the form of a conjecture in Chapter 3: 'The expression complexity of global model checking of a K_t-formula is non-elementary in terms of the alternating box rank of the formula.'

Natural questions that arise from this study: 'What is the strongest modal language for which model checking is decidable on rational Kripke models?' 'Are there other modal operators which preserve regularity but are not definable in the hybrid extension of K_t that can be added to the language?' Identification of such modalities will motivate model checking on rational Kripke models even further.

On the other hand, an important task would be to identify natural subclasses of rational Kripke models where model checking of K_t extended with reachability is decidable.

Concerning the model constructions on rational Kripke frames, the following two questions still need to be answered:

1. Is the unfolding of a rational Kripke frame a rational Kripke tree? (This question was only answered for finite Kripke frames.)

2. Given a pointed rational Kripke model $\mathcal{M}$, is $\mathcal{M}$ finite up to bisimulation? This problem is semi-decidable from the Paige-Tarjan algorithm. I conjecture that this problem is at most semi-decidable. However, this must still be proved.

Rational Kripke trees have decidable first-order theories. As a result model checking tasks of formulas from $H_t(U, \downarrow, \exists)$ is decidable for rational Kripke tree models. However, the follow question still needs to be answered: 'If $\mathcal{M}$ is a rational Kripke tree model, is the set $[\![\varphi]\!]$ for $\varphi \in H_t(U, \downarrow, \exists)$ a regular language?'

Rational Kripke models have been generalised to ω-rational Kripke models in this dissertation. They can now also be generalised to tree rational and ω-tree rational Kripke models.

Finally, is it the case that when a modal formula φ is satisfiable in a rational Kripke frame (i.e. there exists a valuation and state where it is true) then there exists a rational valuation satisfying φ?

Bibliography

[1] S.B. Akers. Binary decision diagrams. *IEEE Transactions on Computers*, C-27(6):509–516, June 1978.

[2] J.A. Anderson. *Discrete Mathematics with Combinatorics*. Prentice Hall, 2001.

[3] J.-M. Autebert and L. Boasson. Transductions rationelles. *Ed. Masson*, 1988.

[4] Y.M. Barzdin and B.A. Trakhtenbrot. *Finite Automata, Behavior and Synthesis*. North-Holland publishing company, Amsterdam, 1973. Translated form the Russian by D. Louvish, English translation edited by E. Shamir and L. H. Landweber, Fundamental Studies in Computer Science, Vol. 1.

[5] M.P. Béal and O. Carton. Determinization of transducers over infinite words: the general case. *Theory Comput. Syst.*, 37(4):483–502, June 2004.

[6] W. Bekker and V. Goranko. Symbolic model checking of tense logics on rational kripke models. In *The Proceedings of the* 5^{th} *Workshop "Methods for modalities" (M4M5)*. Electronic notes of Theoretical Computer Science, 2007. To appear.

[7] J. Berstel. *Transductions and Context-Free Languages*. Teubner Studienbücher, Stuttgart, 1979.

[8] A. Biere, E.M. Clarke, A. Cimatti, O. Strichman, and Y. Zhu. *Bounded model checking*, volume 58, chapter 3, pages 118–149. Academic Press, 2003.

[9] A. Biere, E.M. Clarke, A. Cimatti, and Y. Zhu. Symbolic model checking without BDDs. In *TACAS '99: Proceedings of the 5th International Conference on Tools and Algorithms for Construction and Analysis of Systems*, pages 193–207, London, UK, 1999. Springer-Verlag.

[10] P. Blackburn. Tense, temporal reference, and tense logic. *Journal of Semantics*, 11:83–101, 1994.

[11] P. Blackburn. Representation, reasoning, and relational structures: a hybrid logic manifesto. *Logic Journal of the IGPL*, 8(3):339–625, 2000.

[12] P. Blackburn, M. de Rijke, and Y. Venema. *Modal logic*. Cambridge University Press, New York, NY, USA, 2001.

[13] P. Blackburn and J. Seligman. Hybrid languages. *Journal of logic, language and information*, 4(3):251–272, 1995.

[14] P. Blackburn and J. Seligman. What are hybrid languages? In M. de Rijke, M. Kracht, H. Wansing, and M. Zakharyaschev, editors, *Advances in modal logic*, volume 1, pages 41–62. CSLI Publications, Stanford University, 1998.

[15] A. Blumensath. Automatic structures. Diploma thesis, RWTH-Aachen, 1999.

[16] A. Blumensath and E. Grädel. Automatic structures. In *LICS '00: Proceedings of the 15th Annual IEEE Symposium on Logic in Computer Science*, pages 51–62, Washington, DC, USA, 2000. IEEE Computer Society.

[17] A. Blumensath and E. Grädel. Finite presentations of infinite structures: Automata and interpretations. *Theory of Computing Systems*, 37:641 674, 2004.

[18] L. Boasson and M. Nivat. Adherences of languages. *Journal of Compt. System Sci*, 20(3):285–309, 1980.

[19] A. Bouajjani, J. Esparza, and O. Maler. Reachability analysis of pushdown automata: Application to model-checking. In A. Mazurkiewicz and J. Winkowski, editors, *Proc. of CONCUR '97*, volume 1243 of *LNCS*, pages 135–150. Springer, 1997.

[20] A. Bouajjani, B. Jonsson, M. Nilsson, and T. Touili. Regular model checking. In *CAV '00: Proceedings of hte 12th International Conference on Computer aided verfication*, pages 403–418, London, UK, 2000. Springer-Verlag.

[21] C. Brink. Boolean modules. *Journal of Algebra*, 72(2):29–313, 1981.

[22] C. Brink, K. Britz, and R.A. Schmidt. Peirce algebras. Research Report MPI-I-92-229, Max-Planck-Institut für Informatik, Im Stadtwald, D-66123 Saarbrücken, Germany, July 1992.

[23] J.R. Büchi. Weak second-order arithmetic and finite automata. *Z. Math. logik Grundl. Math.*, 6:66–92, 1960.

[24] J.R. Büchi. On a decision method in restricted second-order arithmetic. In E. Nagel et al., editor, *Logic, Methodology and Philosophy of Science*, pages 1–11. Stanford University Press, Stanford, 1962.

[25] S. Burckel. Functional equations associated with congruential functions. *Theoretical Computer Science*, 123(2):397–406, 1994.

[26] G.W. Cannon, D.S.A. Epstein, D.F. Holt, S.V. Levy, M.S. Paterson, and W.P. Thurston. *Word Processing in Groups*. A.K. Peters, Ltd., Natick, MA, USA, 1992.

[27] A. Carayol and C. Morvan. On rational trees. In Zoltán Ésik, editor, *CSL*, volume 4207 of *Lecture Notes in Computer Science*, pages 225–239. Springer, 2006.

[28] D. Caucal. On infinite transition graphs having a decidable monadic theory. In *ICALP '96: Proceedings of the 23rd International Colloquium on Automata, Languages and Programming*, pages 194–205, London, UK, 1996. Springer-Verlag.

[29] D. Caucal and T. Knapik. A Chomsky-like hierarchy of infinite graphs. In *MFCS '02: Proceedings of the 27th International Symposium on Mathematical Foundations of Computer Science*, pages 177–187, London, UK, 2002. Springer-Verlag.

[30] E.M. Clarke and E.A. Emerson. Design and synthesis of synchronization skeletons for branching time temporal logic. In *Proceedings of the Workshop of logic of programs*, volume 131 of *LNCS*, pages 52–71, Yorktown Heights, New York, USA, 1981. Springerverlag.

[31] E.M. Clarke and E.A. Emerson. Using branching time logic to synthesize synchronization skeletons. *Science of computer programming*, 2:241–266, 1982.

[32] E.M. Clarke and B.H. Schlingloff. Model checking. In A. Robinson and A. Voronkov, editors, *Handbook of automated reasoning*, chapter 24, pages 1635–1790. Elsevier Science Publishers, 2002.

[33] L. Collatz. On the origin of the (3n+1)-problem. volume 3 of *12*, pages 9–11. Journal of Qufu Normal University, 1986. (Chinese, transcribed by Zhi-Ping Ren)[German translation: Über den Ursprung des (3n+1)-Problems, by Zhang-Zheng Yu 1991, Hamburg].

[34] J.H. Conway. Unpredictable iterations. In *Proceedings of the Number Theory conference*, pages 49–52, University of Colorado, Boulder, 1972.

[35] L.M. de Moura, H. Rueß, and M. Sorea. Lazy theorem proving for bounded model checking over infinite domains. In *CADE-18: Prodeedings of the 18th International Conference on Automated Deduction*, volume 2392 of *LNCS*, pages 438–455, London, UK, July 2002. Springer-Verlag.

[36] S. Eilenberg. *Automata, Languages and Machines*, volume A. Academic Press, New York, 1974.

[37] C. Elgot and J. Mezei. On relations defined by finite automata. *IBM Journal of Research and Development*, 9:47–68, 1965.

[38] J. Esparza. An automata-theoretic approach to software verification. In *Developments in Language Theory*, volume 2710 of *LNCS*, page 164, 2003.

[39] J. Esparza and P. Jančar. Deciding finiteness of Petri nets up to bisimulation. In *ICALP '96: Proceedings of the 23rd International Colloquium on Automata, Languages and Programming*, pages 478–489, London, UK, 1996. Springer-Verlag.

[40] J. Esparza, A. Kucera, and S. Schwoon. Model-checking LTL with regular valuations for pushdown systems. In *Proc. of TACS '2001*, volume 2215 of *Lecture Notes in Computer Science*, pages 306–339, 2001.

[41] O. Finkel. On infinitary rational relations and Borel sets. In *DMTCS*, pages 155–167, 2003.

[42] O. Finkel. Undecidability of topological and arithmetical properties of infinitary rational relations. *Informatique Théorique et Applications*, 37(2):115–126, 2003.

[43] M.J. Fisher and M.O. Rabin. Super-exponential complexity of Presburger arithmetic. In *Proceedings of the SIAM-AMS Symposium in Applied Mathematics*, volume 7, pages 27–41, 1974.

[44] C. Frougny and J. Sakarovitch. Synchronized rational relations of finite and infinite words. *Theoretical Computer Science*, 108(1):45–82, 1993.

[45] F. Gire and M. Nivat. Relations rationnelles infinitaires. *Calcolo*, 21(2):91–125, June 1984.

[46] V. Goranko. Infinite state model checking in modal logic. In Prakash Panangaden, editor, *Proceedings of the Twentieth Annual IEEE Symp. on Logic in Computer Science, LICS 2005*. IEEE Computer Society Press, June 2005. Short Presentation.

[47] V. Goranko and M. Otto. Model theory of modal logic. In P. Blackburn, F. Wolter, and J. van Benthem, editors, *Handbook of Modal Logic*, pages 255–325. Elsevier, 2006.

[48] B.R. Hodgson. On direct products of automaton decidable theories. *Theoretical Computer Science*, 19:331–335, 1982.

[49] P. Jančar. Undecidability of bisimilarity for Petri nets and some related problems. *Theoretical Computer Science*, 148(2):281–301, 1995.

[50] P. Jančar and J. Srba. Undecidability results for bisimilarity on prefix rewrite systems. In *Proceedings of 9th International Conference on Foundations of Software Science and Computation Structures (FOSSACS'06)*, volume 3921 of *LNCS*, pages 277–291. Springer-Verlag, 2006.

[51] J.H. Johnson. Rational equivalence relations. *Theoretical Computer Science*, 47(3):39–60, 1986.

[52] B. Jonsson and M. Nilsson. Transitive closures of regular relations for verifying infinite-state systems. In *TACAS '00: Proceedings of the 6th international conference on tools and algorithms for construction and analysis of systems*, pages 220–234, London, UK, 2000. Springer-Verlag.

[53] Y. Kesten, O. Maler, M. Marcus, A. Pnueli, and E. Shahar. Symbolic model checking with rich assertional languages. *Theoretical Computer Science*, 256(1-2):93–112, 2001.

[54] B. Khoussainov and A. Nerode. Automatic presentations of structures. In *LCC '94: Selected Papers from the International Workshop on Logical and Computational Complexity*, pages 367–392, London, UK, 1995. Springer-Verlag.

[55] B. Khoussainov, A. Nies, S. Rubin, and F. Stephan. Automatic structures: Richness and limitations. In *LICS '04: Proceedings of the 19th Annual IEEE Symposium on Logic in Computer Science (LICS'04)*, pages 44–53, Washington, DC, USA, 2004. IEEE Computer Society.

[56] B. Khoussainov, S. Rubin, and F. Stephan. Automatic linear orders and trees. *ACM Trans. Comput. Logic*, 6(4):675–700, 2005.

[57] T. Kropf, R. Kumar, and K. Schneider. Alternative proof precedures for finite-state machines in higher-order logic. In *HUG '03: Proceedings of the 6th international Workshop on higher-order logic theorem proving and its applications*, pages 213–226, London, UK, 1994. Springer-Verlag.

[58] G.M. Kuper and M.Y. Vardi. On the complexity of queries in the logical data model. In *Lecture notes in computer science on ICDT '88*, pages 267–280, New York, NY, USA, 1988. Springer-Verlag New York, Inc.

[59] O. Kupferman, M. Y. Vardi, and P. Wolper. An automata-theoretic approach to branching-time model checking. *Journal of the ACM*, 47(2):312–360, March 2000.

[60] C.Y. Lee. Representation of swithching circuits by binary decision programs. *Bell Systems Techinical Journal*, 38:985–999, 1959.

[61] A.H. Lewis. An introduction to Petri nets. George Mason University, 1999. http://ict.tippinst.ie/%7Ejhannafin/RandD/petri-intro.pdf.

[62] R. Lindner and L. Staiger. Algebraische codierungsteorie -theorie der sequentiellen codierunge. *Akademie-Verlag*, 1977.

[63] N. Markey. Temporal logics. Course notes, Master Parisien de Recherche en Informatique, Paris, France, 2006.

[64] R. McNaughton. Testing and generation infinite sequences by a finite automaton. *Information and Control*, 9:521–530, 1966.

[65] C. Meinel and T. Theobald. *Algorithms and Data Structures in VLSI Design: OBDD - Foundations and Applications*. Springer-Verlag, Berlin, 1998.

[66] A. Meyer. *Finitely presented infinite graphs*. Thèse de doctorat, University of Rennes, October 2005.

[67] A. Moen. Introduction to Petri nets, 2003. Lecture notes, http://www.ifi.uio.no/dbsem/foils2003vaar/.

[68] B.M.E Moret. Decision trees and diagrams. *ACM Computing Surveys*, 14(4):593–623, 1982.

[69] C. Morvan. On rational graphs. Rapport de recherche 3944, INRIA, Mai 2000.

[70] C. Morvan. *Les graphes rationnels*. Thèse de doctorat, Universit de Rennes 1, 2001.

[71] C. Morvan. Classes of rational graphs. http://www.irisa.fr/JM06/Papers/morvan.pdf, 2006.

[72] D.E. Muller. Infinite sequences and finite machines. In *Proc.* 4^{th} *IEEE Symp. on Switching Circuit Theory and Logical Disign*, pages 3–16, 1963.

[73] M. Nilsson. *Regular model checking*. Phd. thesis, Uppsala University, Uppsala, Sweden, March 2005.

[74] M. Nivat. Transductions des langages die Chomsky. *Ann. Inst. Fourier (Grenoble)*, 18:339–456, 1968. In French.

[75] D. Perrin and J.E. Pin. *Infinite words*, volume 141 of *Pure and Applied Mathematics*. Elsevier, 2004. ISBN 0-12-532111-2.

[76] M. Presburger. Über die vollstaendigkeit eines gewissen systems der arithmetik ganzer zahlen in welchem die addition als einzige operation hervortritt. *Comptes Rendus du I congrés de Mathematiciens des Pays Slaves*, pages 92–101, 1929.

[77] M.O. Rabin. Decidability of second-order theories and automata on infinite trees. *Trans. Amer. Math. soc.*, 141:135, 1969.

[78] C. Rispal. The synchronized graphs trace the context-sensitive languages. In *Proceedings of the 4th International Workshop on Verification of Infinite-State Systems (Infinity 2002)*, volume 68. Electronic notes of Theoretical Computer Science, 2002.

[79] S. Rubin. *Automatic Structures*. PhD thesis, The University of Auckland, 2004.

[80] N. Sântean. Memento transducers. *Reading Course Presentation,CS 900 Readings in Computer Science*, 2002.

[81] K. Schneider and T. Schuele. Bounded model checking of infinite state systems: exploiting the automata hierarchy. In *Proceedings of the second ACM and IEEE International Conference on Formal Methods and Models for Co-Design, (MEMOCODE)*, pages 17–26, San Diego, CA, USA, June 2004. IEEE Computer Society.

[82] K. Schneider and T. Schuele. Bounded model checking of infinite state systems. *Form. Methods Syst. Des.*, 30(1):51–81, 2007.

[83] C. Stirling. The joys of bisimulation. In *MFCS '98: Proceedings of the 23rd International Symposium on Mathematical Foundations of Computer Science*, pages 142–151, London, UK, 1998. Springer-Verlag.

[84] T.A. Sudkamp, editor. *Languages and machines : an introduction to the theory of computer science*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1988.

[85] W. Thomas. Languages, automata and logic. In *Handbook of formal languages, vol. 3: beyond words*, pages 389–455. Springer-Verlag New York, Inc., New York, NY, USA, 1997.

[86] W. Thomas. A short introduction to infinite automata. In *DLT '01: Revised Papers from the 5th International Conference on Developments in Language Theory*, pages 130–144, London, UK, 2002. Springer-Verlag.

[87] W. Thomas. Constructing infinite graphs with a decidable MSO-theory. In B. Rovan and P. Vojtàs, editors, *MFCS*, volume 2747 of *Lecture Notes in Computer Science*, pages 113–124. Springer, 2003.

[88] Tanguy Urvoy. Regularity of congruential graphs. In *MFCS '00: Proceedings of the 25th International Symposium on Mathematical Foundations of Computer Science*, pages 680–689, London, UK, 2000. Springer-Verlag.

[89] J. van Benthem. Modal logic and classical logic. *Bibliopolis*, 1983.

[90] M.Y. Vardi. An automata-theoretic approach to linear temporal logic. In F. Moller and G. Birtwistle, editors, *Logics for Concurrency: Structure versus Automata*, volume 1043 of *Lecture Notes in Computer Science*, pages 238–266. Springer-Verlag, Berlin, 1996.

[91] Y. Venema. Temporal logic. In L. Globe, editor, *The Blackwell Guide to Philosophical Logic*, pages 203–223. Blackwell Publishers, 2001.

[92] I. Walukiewicz. Model checking CTL properties of pushdown systems. In *FSTTCS*, pages 127–138, 2000.